Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webová analýza a Eye Tracking

Webpage Analysis and Eye Tracking

2015 **Bc. Jan Vykopal**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jan Vykopal**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Webová analýza a Eye Tracking
Webpage Analysis and Eye Tracking

Zásady pro vypracování:

Cílem práce je integrovat tehnologii Eye tracking do prostředí webových stránek a umožnit tak přímou analýzu chování uživatele.

1. Zmapujte a popište možnosti a způsoby zachytávání a zpracování pohledů uživatele při práci na počítači.
2. Navrhněte způsob analýzy jeho chování na základě pohledu, a to přímo v prostředí webového prohlížeče resp. webové stránky.
3. S využitím systému GazePoint i implementujte systém, který umožní analýzy chování při práci na webu. Zohledněte sémantické prvky a osbahovou strukturu webových stránek.
4. Zhodnoťte výsledné řešení a jeho použitelnost při reálných úkolech analýzy UX.

Seznam doporučené odborné literatury:

[1] K. Holmqvist, M. Nystrom: Eye Tracking, Oxford University Press, 2011, ISBN: 978-0-19-969708-3
[2] J. Nielsen: Designing Web Usability, New Riders Pub., 2000, ISBN: 978-1-56205-810-4

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Radecký, Ph.D.**

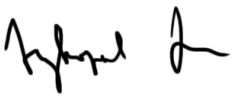
Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.



V Ostravě dne: *3. května 2015* podpis studenta

Poděkování

Rád bych poděkoval Ing. Michalu Radeckému, Ph.D. za jeho odbornou pomoc a konzultaci při vytváření této diplomové práce a také všem, kteří se stali testovacími uživateli a pomohli tak k ověření funkčnosti řešení.

Abstrakt

Alespoň nějaké využití webové analýzy, je považováno za samozřejmost u každého moderního veřejného webu. V kombinaci s technologií sledování pozice pohledu na obrazovce, lze dosáhnout zajímavých náhledů na uživatelovo chování při procházení webové stránky. Pro identifikaci konkrétních sémantických prvků, je ale nutné často komplikované a zdlouhavé manuální srovnávání. Aby tato kombinace podala všechny důležité informace automaticky, upravil jsem v této práci způsob, jakým probíhá zaznamenávání interakce se sémantikou webové stránky. Je tak dosáhnuto pohledu, který je schopen určit, jak dané části zdrojového kódu zaujímají pozornost měřených uživatelů při průchodu webovou stránkou, respektive při plnění specifického úkolu. Pro měření uživatelova pohledu a následné testování řešení jsem použil zařízení Gazepoint GP3 Eye Tracker, které mi pro tento účel bylo zapůjčeno VŠB-TU Ostrava.

Klíčová slova

Webová analýza; analýza; Sledování pohledu očí; Gazepoint; Eye tracker; GP3; Webová stránka; Sémantika; Web sockets; WinJs; HTML5

Abstract

At least some usage of web page analysis is considered necessary for every modern public website. In combination with eye tracking technology, it is possible to discover interesting insights into the user's behavior, while he is browsing measured website. For purposes of identification concrete semantics elements of underlying structure, it is often necessary to manually compare measured information and that can be difficult and time consuming. To automate this procedure I modified the way of recording user's interaction with webpage semantics. Because of resulting view, it is possible to get information on which parts of source code are getting more user's attention, while user browsing through a page or completing specific tasks. The device used for eye tracking was Gazepoint GP3, which was loaned to me by the VŠB - TU Ostrava.

Key words

Web analysis; Analysis; Eye tracking; Gazepoint; Eye tracker; GP3; Web page; Semantics; Web sockets; WinJs; HTML5

Seznam použitých zkratek

Zkratka	Význam
AJAX	Asynchronous JavaScript and Extensible Markup Language, Asynchronní Javascript a rozšiřitelný značkovací jazyk
API	Application Programming Interface, Aplikační rozhraní pro programování
BPOG	Best position of gaze, Nejlepší pozice pohledu
CMOS	Complementary Metal–Oxide–Semiconductor, Doplňující se kov-oxid-polovodič
CR	Cornea reflection, Odraz rohovky
CRISP-DM	Cross Industry Standard Process for Data Mining, Mezi průmyslový standartní proces pro dolování dat
CSS	Cascading Style Sheets, Kaskádové styly
CSV	Comma-Separated Values, Hodnoty rozdělené čárkou
DM	Data mining, Dolování dat
DOM	Document Object Model, Objektový model dokumentu
EOG	Electrooculography, Elektrookulografie
FPS	Frames per second, Snímků za vteřinu
GUID	Globally Unique Identifier, Globálně unikátní identifikátor
HCI	Human-computer interaction, Interakce mezi počítačem a člověkem
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ICC	International Carpathian Control
IETF	Internet Engineering Task Force, Komise techniky Internetu
IP	Číslo, které jednoznačně identifikuje síťové rozhraní v počítačové síti, která používá IP (internetový protokol)
IR	Infrared, Infračervené
JSON	JavaScript Object Notation, JavaScriptový objektový zápis
LED	Light-Emitting Diode, Dioda emitující světlo
RFC	Request For Comments, Žádost o komentáře
RGB	Color model red-green-blue, Barevný model červená-zelená-modrá

RIA	Rich Internet Application, Bohatá internetová aplikace
SEMMA	Sample, Explore, Modify, Model and Assess
SEO	Search Engine Optimization, Optimalizace pro vyhledávače
SHA1	Secure Hash Algorithm
SPA	Single page application, Jendostránková aplikace
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol, Primární přenosový protokol/protokol síťové vrstvy
UI	User interface, Uživatelské rozhraní
URL	Uniform Resource Locator, Jednotná adresa zdroje
USB	Universal Serial Bus, Univerzální sériová sběrnice
UTF8	Universal Character Set (Univerzální znaková sada) + Transformation Format (Transformační formát) + 8 bit (velikost 8 bitů)
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language

Obsah

Úvod

V současné době není pochyb o tom, že webové stránky jsou podstatnou součástí nejen internetu jako technologie, ale i životů milionů lidí po celém světě. S velkou popularitou jde ruku v ruce snaha analyzovat chování uživatelů a webových stránek a tím lépe přizpůsobit software těm, pro které je určen. Jednou z hlavních motivací této práce je poskytnout nástroj pro analýzu chování uživatelů na konkrétních webových stránkách. Ačkoli již existuje mnoho různých nástrojů a technik pro analýzu webových stránek (např. Google Analytics), v tomto případě jde o využití technologie pro sledování pohledu očí na obrazovce počítače (tzv. Eye tracking). Použití podobných technologií jako nástroje analýzy chování na webu není nic nového, nicméně využití nových technologií jako je eye tracking umožňuje podrobnější pohled na danou problematiku v kombinaci se sémantikou zdrojového kódu webové stránky.

Jedním z benefitů tohoto přístupu je možnost vylepšení použitelnosti uživatelského rozraní. Problémem často je, že kvalitu uživatelského rozhraní není možné jednoduše kvalifikovat nebo hodnotit. Mnozí, aspekty jako je tento označují druhem umění, jehož kvalifikace a hodnocení je také nejednoznačné. Oproti klasickým druhům umění čelí dobré uživatelské rozhraní tomu, že jeho hlavním účelem je funkčnost, ne estetika, která poskytuje pouze přidanou hodnotu.

S použitím vizualizace a nahrávání chování uživatele na testovacích stránkách v reálném čase tato práce nabízí jeden z možných pohledů na hodnocení použitelnosti uživatelského rozhraní. Schopnost rozhodnout, co jsou silné a slabé stránky uživatelského rozhraní, přímo na základě chování uživatele má velký potenciál, protože nyní je tato schopnost často svázána s použitím periferií jako je myš nebo klávesnice a tak přímo nemusí reprezentovat okamžité pocity uživatele. S využitím technologie sledování očí je možné zjistit přímo, které části webové stránky zaujmuly nejvíce uživatelovu pozornost a kterým se naopak uživatel vůbec nevěnoval, taková informace je zajímavá například pro marketingové využití.

Tato práce je rozdělena do několika částí a postupně popisuje použité řešení a jeho výsledky. Pro veškerá měření byl použit přístroj GazePoint GP3 Eye Tracker.

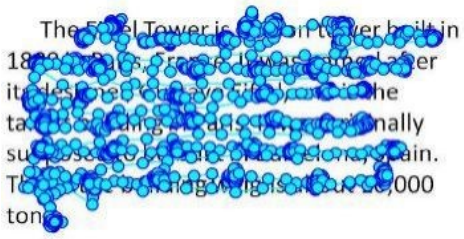
1 Eye tracking

1.1 Co je eye tracking

Eye tracking je často popisován jako proces měření pohledu očí nebo jako pohyb očí relativní vůči hlavě. Protože člověk nesleduje jenom jeden bod, pojmem eye tracking je často myšlen středový bod ostrého pohledu uživatele. Pro zaznamenání pohledu očí se využívají zařízení, běžně označovaná pojmem eye tracker. První pokusy o vytvoření takovýchto zařízení se objevovaly již během 19. století. Tato zařízení byla převážně mechanická, složitá a příliš invazivní vůči měřeným uživatelům, proto nevyhovující. Až v roce 1901 byl Dodgem a Clinem představen princip měření pomocí fotografování odrazu externího zdroje světla a fovea centralis (tj. oblast oka zodpovědná za ostré centrální vidění). Tento přístup byl mnohem méně invazivní než ty předešlé a později se rozvinul v nejpopulárnější techniku zaznamenávání pohledu očí. Od druhé poloviny 20. století se začaly objevovat další alternativní techniky pro eye tracking. Jednou z takových byla technika použití systému čoček a zrcadel nabízející velký detail měřených dat, ale její použití bylo nepohodlné. Odlišně k problému přistupovaly systémy využívající měření elektromagnetické indukce v silikonové kontaktní čočce vložené do oka při anestézii. Tyto systémy byly po dlouhou dobu považovány za ty nejpřesnější, ale dnes je známo, že ovlivňují chování očí svého nositele. Dalším alternativním přístupem bylo měření elektromagnetické variace v místech pohybových svalů oka (tzv. EOG). EOG často měřilo pouze horizontální směr pohybu a trpělo rušením z okolních svalů. EOG se stále používá jako levná alternativa modernějších systémů, ale i přes velkou vzorkovací frekvenci zůstává nepřesnou technikou měření pohybu očí. Tyto a další techniky záznamu znamenaly konstrukci vlastních zařízení, což sice zpomalovalo a komplikovalo výzkum, ale výhodou byla znalost konkrétních zařízení a jeho funkce u těch, kteří je používali. [1] Dnes existuje možnost nákupu takovýchto zařízení, ale kvůli jejich relativně malému počtu a nízkému stupni otestování existuje pořád riziko chyby nebo neočekávané nepřesnosti zařízení. Jakmile se tyto zařízení dostanou do početnější produkce, bude se zvyšovat i šance na objevení jejich případných nedostatků nebo chyb a tím se nepochybně bude zvyšovat i jejich kvalita.

Ačkoli eye tracking byl dlouhou dobu využíván převážně v akademickém prostředí (tato skupina uživatelů je pořád tou nejčetnější), využití technologií sledování pohledu jsou častou součástí marketingových výzkumů a průzkumů. Zařízení v takových případech slouží převážně jako nástroj pro informovaná rozhodování týkající se marketingové kampaně. Další podstatné využití je v medicíně a souvisejících výzkumech. Zařízení mohou být například využívány při operacích očí. Velkou skupinou je použití pro ovládání respektive interakci s počítačem (human-computer interaction, HCI). Tento způsob ovládání je využíván uživateli, jejichž onemocnění nebo stav znemožňuje či znesnadňuje ovládání počítače klasickými způsoby. Mimo zmíněné lze využití eye trackingu nalést v psychologických studiích, neurologických výzkumech nebo automobilovém průmyslu. Spektrum využití je rozsáhlé a s lepší dostupností zařízení budou další oblasti použití rychle přibývat. V této práci se převážně zabývám využitím

technologie eye tracking pro práci a analýzu uživatelských rozhraní webových stránek s ohledem na sémantiku zdrojového kódu.



Obrázek 1.1: Ukázka použití sledování pohledu při analýze četby textu

1.2 Stav technologie

V současné době existuje několik různých způsobů jak zaznamenat pozici pohledu očí a tím pádem i několik různých druhů zařízení pro tento záznam.

Podle způsobu snímání lze zařízení rozdělit do tří hlavních skupin. Jde o použití speciálních kontaktních čoček, měření elektrického potenciálu a optické měření. Nejčastěji preferovanou metodou je optické měření, protože jde o levný a neinvazivní způsob s často dostatečnou kvalitou získaných dat.

I když je většina moderních zařízení optického (resp. video záznamového) typu založena na měření odrazu zornice a rohovky oka, samotné zařízení je možné rozdělit do několika kategorií. [1] [2]

- Nejběžnější jsou statická zařízení, které mají jak osvětlující, tak snímací kameru na stole před měřeným uživatelem. Buď jde o zařízení nějakým způsobem spojené s hlavou uživatele (omezení pohybu hlavy uživatele) nebo jde o zařízení volně postavené před uživatele (do této kategorie patří například použité zařízení Gazepoint GP3).
- Mírně odlišný přístup zaujímají zařízení, připojená přímo k hlavě uživatele a to pomocí speciální helmy (čepice, brýlí, apod.), na které jsou opět připevněny jak osvětlovací, tak snímací kamery.
- Třetím typem je zařízení, připojené k hlavě a obohacené o další kameru snímající pozici hlavy v okolní scéně (tzv. head tracker).

Za zmínku určitě stojí, že existuje několik různých výzkumů slibujících možnosti sledování pozice očí i pomocí webové kamery bez dodatečných hardwarových úprav (např. společnost xLabs) nebo pomocí kamery na mobilních zařízeních. Další slibnou možností je využití běžně dostupných zařízení, které jsou původně určené pro ovládání videoher (např. Microsoft Kinect). I když tyto nové způsoby ještě nemusí dodávat data o takové přesnosti jako dedikované eye tracker zařízení, jejich vývoj slibuje mnohem větší potencionální základnu uživatelů a tím pádem i více možných měření. [3]

1.3 Zařízení Gazepoint GP3 Eye Tracker

Zařízení eye tracker použité pro měření v této práci je Gazepoint GP3 Eye Tracker. Jde o třetí generaci samostatného stolního zařízení od společnosti Gazepoint.

Zařízení obsahuje Firefly MV mono USB 2.0 kameru s mikro čočkami a infračerveným (IR) filtrem, kompaktní infračervené LED světlo. Vše je chráněno pomocí 3D vytisknutého krytu. Vstupní a výstupní konektory jsou reprezentovány dvěma USB porty (jeden pro napájení a druhý pro data). [4]

Použitá 0.3MPx kamera obsahuje 1/3" Micron MR9V022 CMOS obrazový senzor s velikostí pixelu 6 µm x 6 µm, globální uzávěrku a rychlostí snímaní 60 snímků za sekundu (fps) při rozlišení 752px x 480px. Navíc zařízení pomocí USB 2.0 rozhraní nabízí 8 a 16 bitový datový výstup. [4]

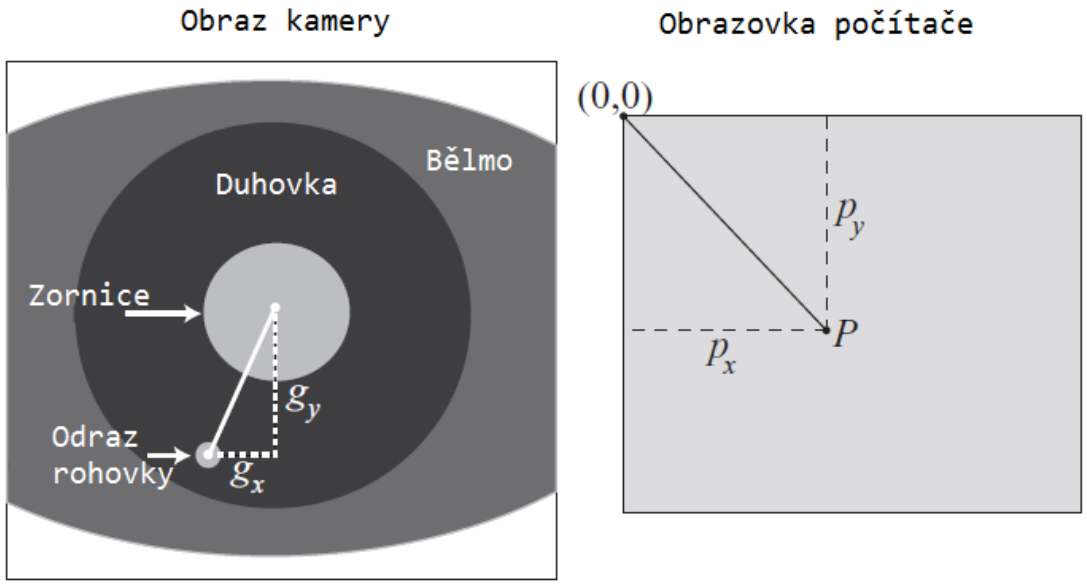
Ovládací software poskytuje aplikační rozhraní (API) pro nastavení požadovaných parametrů. Data jsou pak posílána ve formě datového proudu na adrese 127.0.0.1 a portu 4242 pomocí protokolu TCP/IP. Posílané data jdou ve specifikovaném formátu XML.

Software zařízení obsahuje mimo jiné i nástroj pro kalibraci zařízení pomocí pěti nebo devíti bodů. Zařízení je schopno pracovat i s mírnými pohyby hlavy měřeného uživatele do 25cm horizontálně, 11cm vertikálně a při ±15 centimetrových změnách ve vzdálenosti od zařízení. Udávaná frekvence celého zařízení je 60Hz. GP3 eye tracker je kompatibilní s obrazovkami do velikosti úhlopříčky 24" včetně.

Obrázek 1.2: Zařízení Gazepoint GP3 Eye Tracker

1.4 Jak to funguje

U většiny zařízení pro eye tracking se kamera zaměří na jedno nebo obě oči měřeného uživatele a nahraje jejich pohyb během prováděného testu. Zařízení používají střed zornice a infračervené (nebo téměř infračervené) nerovnoběžné světlo k vytvoření odrazu rohovky (CR). Vektor mezi centrem zornice a odrazem rohovky se může použít k vypočítání bodu pohledu. Jednoduchá kalibrační procedura umožňuje přepočítávání získaného bodu z kamery na pozici obrazovky.

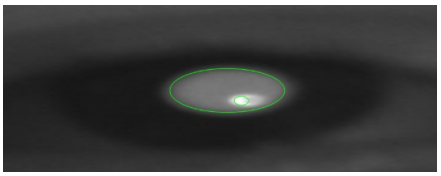


Obrázek 1.3: Grafické znázornění způsobu výpočtu pozice pohledu zařízením (hodnoty z obrazu kamery se přepočítají na adekvátní hodnoty na obrazovce) [5]. Hodnoty g ze snímku oka se pomocí kalibrační procedury přepočítají na hodnoty p (pixelové pozice na obrazovce počítače). Konkrétní vlastnosti přepočtu ovlivňuje použitá obrazovky, vzdálenost a pozice uživatele uživatele i pozice zařízení pro eye tracking. Proto je pozice vypočítávána pomocí individuální kalibrace před každým měřením.

Dvě techniky využívající infračervené (nebo téměř infračervené) světlo jsou techniky využívající světlou a tmavou reprezentaci zornice. Rozdíl je v použití umístění zdroje osvíceni respektive k optice. Pokud je osvícení koaxiální s optickou cestou (resp. sdílejí společné osy), oko se chová jako odrazové sklo (resp. retroreflektor). Jakmile se světlo odrazí od sítnice, vytvoří světlý obraz zornice (efekt podobný efektu červeného oka, který může vzniknout při fotografování). Pokud je zdroj osvícení posunutý vůči optické cestě, zornice se jeví tmavá, protože odraz ze sítnice je veden mimo kameru. [1]

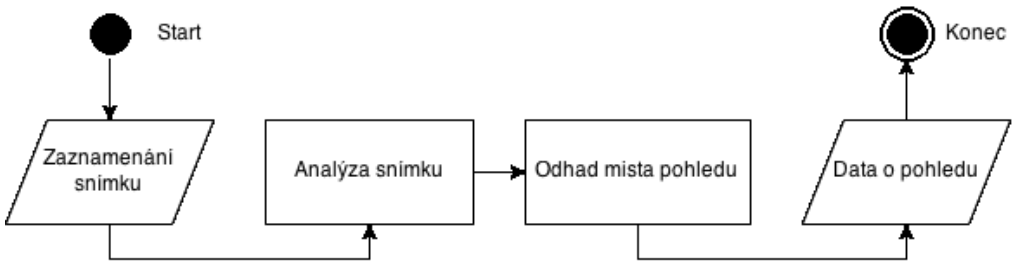
Technika používající světlé zobrazení zornice vytváří větší kontrast mezi duhovkou a zornicí. To umožňuje spolehlivější sledování očí všech pigmentací a výrazně snižuje rušení způsobené očními řasami a jinými překážkami ve směru k oku. Toto nastavení také zvyšuje světelné rozsahy, ve kterých je možné měření provádět. Nevýhodou jsou horší vlastnosti při

měření ve venkovních prostorách. Techniku světlých zobrazení zornic využívá i použité zařízení Gazepoint GP3.



Obrázek 1.4: Ukázka reprezentace oka z používaného zařízení, které využívá techniku světlého zobrazení zornice.

Alternativou k technikám světlé nebo tmavé zornice je metoda pasivního světla. Využívá se světla z viditelného spektra k osvětlení a to může být nepříjemné pro měřeného uživatele. Protože kontrast je při použití této metody menší než při použití některé z metod využívající aktivního světla, využívá se střed duhovky pro výpočet vektoru. Tento výpočet vyžaduje znalost hranice mezi duhovkou a očním bělmem. Kvůli zjevným nevýhodám je použití této metody méně časté. [1] [6]



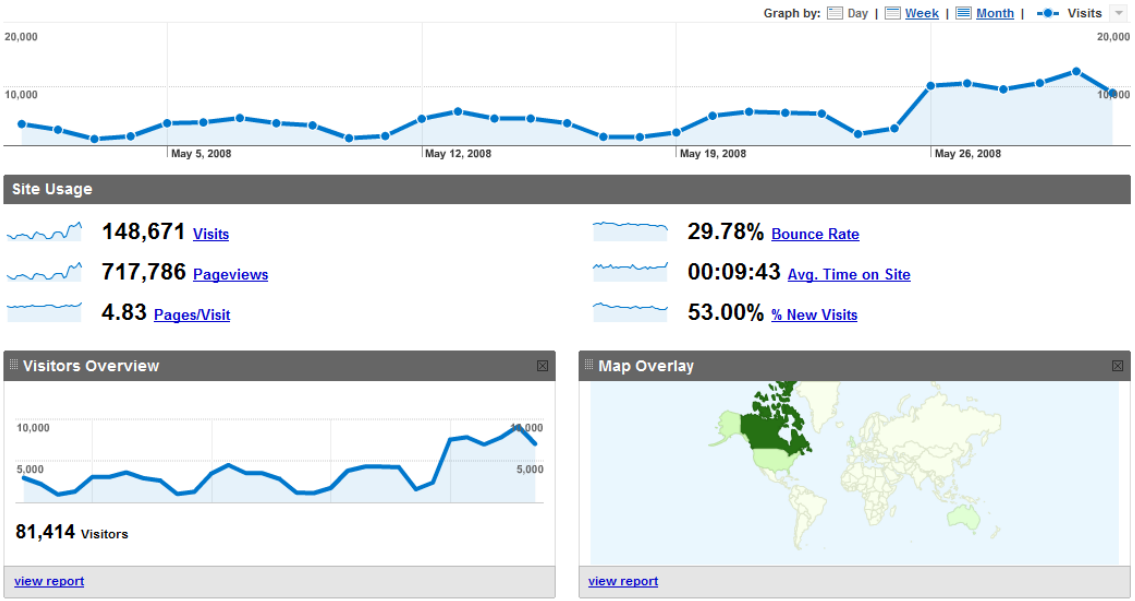
Obrázek 1.5: Zjednodušený pohled na funkcionalitu eye tracking zařízení

2 Webová analýza a Eye Tracking

2.1 Proč to dělat

Nejprve je užitečné ujasnit si, co se rozumí pojmem webová analýza (resp. Web analysis). Existuje několik definic různých definic tohoto pojmu. Definice podle Digital Analytics Association popisuje webovou analýzu jako měření, sběr a reportování internetových dat pro pochopení a optimalizaci webového použití. Webová analýza tedy není jenom nástroj pro měření návštěvnosti webu, ale navíc se snaží četnými nástroji hledat vzory chování, navrhovat možné postupy optimalizace a umožnit tak lepší přístupnost a použitelnost webu nejen pro nové uživatele. Webovou analýzu je tedy možné chápat jako jednu z částí internetového marketingu.

Do skupiny internetového marketingu patří například SEO (Search engine optimization), jenž se snaží o lepší čitelnost webové stránky pro vyhledávače, analýza vyhledávání a různé druhy marketingu včetně emailového marketingu, marketingu přes doporučení, kontextuálního marketingu, mobilního marketingu, marketingu na sociálních sítích a mnoho dalších. Mezi hlavní motivace internetového marketingu tedy spadá například vytvoření přesné reklamy vytvořené pro konkrétního uživatele, na základě jeho chování na síti. Personalizovaná reklama bere v potaz uživatelovy zájmy na internetu (vyhledávač poskytne informace o tom, co uživatel hledá, jaké jsou jeho oblíbené a časté cesty po internetu, kde tráví nejvíce času apod.). Dalším využitím může být reklama na webové stránky samotné (do této podkategorie patří například dříve zmíněné SEO). V podstatě každá webová stránka má dnes na internetu velkou konkurenci a její nalezení není bez využití těchto technik jednoduché, ne-li přímo nemožné.



Obrázek 2.1: Ukázka některých informací, které poskytuje moderní nástroj pro webovou analýzu.

Hlavním záměrem této práce je pomocí využití technologie eye tracking kvalifikovat, měřit a hodnotit uživatelské rozhraní, na základě měření uživatele. Reklama může být také ve specifických případech brána jako forma uživatelského rozhraní, ale já budu nadále pod pojmem uživatelského rozhraní (resp. UI) uvažovat ovládací prvky a obsah webových stránek jako celku. Pro tyto úkoly nabízí webová analýza několik různých nástrojů. Jedním z nich je monitorování chování uživatele a jeho cesty webem, je tak možné určit kvalitu a srozumitelnost daných stránek. Monitorování pozice myši se často provádí s pomocí takzvaných heatmap (tepelných map - teplejší barvy značí větší četnost a studenější barvy četnost menší). Heatmapa barevně rozlišuje ty části webové stránky, které zaujmou uživatele. Populární je i reprezentace zarolování stránek pomocí takzvaných scrollmap. Díky této technice bylo například několikrát zjištěno, že většina uživatelů stránku alespoň mírně zaroluje, i když její obsah není zajímavý. Technik a nástrojů využívaných dnešními nástroji webové analýzy je mnoho a liší se s každým poskytovatelem takových služeb. [7]

Pro zaznamenávání uživatelova chování a vstupů se nejčastěji využívá pozice myši a vstupy uživatele jako je například vyhledávání. Problémem tohoto přístupu je zjevná mezera mezi tím co uživatel sleduje a na co kliká pomocí myši. Sledování očí tuto mezeru minimalizuje a v reálném čase umožňuje relativně přesnou reprezentaci chování uživatele. Eye tracking je tedy nutné brát jako další vstup uživatele, protože myš vždy nereprezentuje uživatelovy pocity z uživatelského rozhraní. S takovouto technologií je potřeba klást si i otázky týkající se bezpečnosti a ochrany soukromí uživatele. Sledování pohledu uživatele může být spojeno i s myšlenkovým postupem uživatele a může tak teoreticky poskytnout nástroj pro evaluaci psychologických vlastností uživatele [8].

V době vzniku této práce zařízení pro sledování pozice očí nejsou běžnou součástí vybavení počítačů a slouží tedy spíše pro výzkumné účely. Ochrana soukromí není tedy tak velkým problémem, jako by mohla být, kdyby se tato technologie stala podobně oblíbenou a běžnou, jako jsou webové kamery a jiné periferie nalezené na běžném notebooku či tabletu. To, že jsou si uživatelé vědomi toho, že je jejich chování nahráváno by mohlo ovlivnit jejich rozhodování a chování na webu. Protože pohyby očí jsou z podstatné části podvědomé, není toto riziko na takové úrovni, aby příliš znehodnocovalo výsledky měření.

Ačkoli nástroje webové analýzy jsou většinou závislé na JavaScriptu a tím pádem na běhu přímo u uživatele, data, získané tímto způsobem nemusí přesná a ani nemusí reprezentovat celou realitu. Jedním z důvodů proč tomu tak je, je použití mnohých skriptů a rozšíření, které práci analytických skriptů znemožňují nebo upravují. Data získané ze skriptu běžícího u uživatele tedy mohou reprezentovat jenom určitý zlomek spektra uživatelů. Nelze například jenom díky výsledkům analytického skriptu usoudit, že stránka není dobře navržená, protože může existovat velké množství uživatelů, u kterých skript neběží, nebo nefunguje jak má. Nicméně to nemusí vlastníka webu příliš znepokojovat, webová analýza se nespoléhá jenom na skripty a i díky tomu je schopna například zjistit navýšení návštěv v určitém období, nárůst kliknutí na daný odkaz a podobně. Taková informace může být dobrý nástroj pro upravení byznys plánu a provádění lepších informovaných rozhodnutí. Často se tyto data také používají pro odůvodnění nějaké změny v designu nebo provedení stránek. Je tedy jasné, že pro validaci

návrhu stránky, nebo jejího designu není samotná webová analýza dostačující. Rozhodně ale není zbytečná a její benefity byly v minulosti dokázány v mnoha různých případech užití [9] [10]. Pro tuto konkrétní práci není blokování skriptů ani rozdílné chování webových prohlížečů podstatné, protože veškeré měření je prováděno v podmínkách, vhodných pro vyvinutý software. Jeho funkčnost tedy nebude omezena, ale v případech volného užití je na zvážení jestli i toto brát v potaz.

Spojení webové analýzy a technologie Eye Tracking pro velký počet uživatelů (myšleno jako běh skriptu na stránce v ostrém provozu) není technicky proveditelný úkol, alespoň co se týče technologií, jako je zmíněno dříve. Propojení webové analýzy a Eye Tracking technologie je tedy myšleno pro testování stránek na malém, testovacím počtu, uživatelů. Protože zařízení sleduje přímo uživatelův pohled, nestojí nic mezi ním a designem (nebo uživatelským rozhraním) stránky. Například pokud uživatel bude dlouho hledat nějaký ovládací prvek (tlačítko v menu, odkaz na stránce a podobně), bude to datech ze zařízení Eye Tracking zřejmé, ale na datech z periferiích, jako jsou myš nebo klávesnice to viditelné být nemusí (a ve většině případů ani není). V současné době již existuje mnoho způsobů propojení technologie sledování pohledu očí a webových stránek, ale neexistuje propojení přímo se zdrojovým kódem stránky a tak i její sémantikou (alespoň ne co jsem si vědom). Toto propojení se sémantikou stránky nabízí možnost přímo zjistit, jak uživatelsky přívětivý a použitelný je design stránek respektive uživatelské rozhraní. Ve výsledku je také viditelné, které části zdrojového kódu zůstávají neviděny a které upoutávají uživatelovu pozornost. Navíc není nutné složité porovnávání výsledků analýzy a částí kódu, kterých se tyto výsledky týkají.

2.2 Získávání dat z webových stránek

Jeden z prvních nápadů jak propojit zařízení na Eye Tracking a webovou stránku spoléhal na přímé spojení s pomocí technologie WebSockets [11]. Jde o obousměrné full duplex spojení, které samotný http protokol neumožňuje. Tím pádem by nebylo nutné mít prostředníka v podobě serveru, který data přeposílá. Tento způsob by byl nejspíše tím nejintuitivnějším a nejjednodušším. Nicméně toto řešení se ukázalo jako technicky nerealizovatelné, protože ovladač zařízení GazePoint GP3 umí posílat svá data jen jako TCP/IP stream. Tento stream nebylo možné jednoduše číst na straně uživatele ve webové stránce, protože protokol WebSockets potřebuje vlastní způsob nastavení komunikace (specifický handshake aj.). Další alternativou by mohlo být řešení pomocí robustní implementace vlastního webového prohlížeče, s využitím některého z dostupných jader prohlížečů (např. projekt Chromium). To by znamenalo speciální aplikaci a nasazení v reálných podmínkách by tak bylo složitější.

Kvůli tomuto problému jsem se uchýlil k řešení pomocí prostředníka v podobě serveru. Ač se to nemusí zdát patrné na první pohled, toto řešení má i několik výhod. Lze například data přetransformovat do paměťově méně náročného formátu JSON (tj. JavaScript Object Notation). Protože se posílá relativně velký počet dat, je výhodnější použít tento formát, než více paměťově náročné XML, které posílá zařízení pro Eye Tracking. Dále prostředník může data validovat a odebírat chybné měření. Středová aplikace také umožňuje odstínit skript od nastavování zařízení, časování a synchronizace a dalších detailů, díky kterým se skript může

soustředit pouze na sběr potřebných dat. To má mimo jiné za důsledek menší velikost skriptu, což nepatrně urychlí načítání testovacích stránek, ale i jejich mírně rychlejší běh. Rozdíl ve výkonu by mohl být zpozorovatelný na méně výkonných počítačích nebo starších mobilních zařízeních. Největší nevýhodou tohoto přístupu je další aplikace, která musí být samostatně spuštěna. Zabírá tak další zdroje hostitelského počítače a komplikuje práci uživateli.

Díky prostředníkovi je aplikace schopna měřit chování uživatele napříč webovými stránkami a není tak omezena na ukládání po každé stránce, i když je to kvůli přehlednosti výstupného souboru doporučeno. Je tak možné sbírat data napříč celým případem užití, bez opakovaného nastavování zařízení pro sledování pohybu očí na obrazovce.

Zařízení poskytuje relativně velké množství různých parametrů a měření. Pro účely této aplikace jsou využity hodnoty pozice myši a takzvaný BPOG. BPOG (the best position of gaze) je nejlepší vybraná pozice pohledu a skládá se ze dvou hodnot BPOGX a BPOGY. Jedna hodnota reprezentuje horizontální pozici pohledu očí (X), druhá reprezentuje vertikální pozici pohledu očí (Y). Alternativou by bylo zaznamenávat levou a pravou část pozice pohledu zvlášť, ale to pro tuhle konkrétní aplikaci není potřeba. Zajímám se jenom o pohled na obrazovku a není tak důležité, kam směřují pohledy jednotlivých očí. Mimo hodnoty BPOG zařízení posílá i binární hodnotu validity měření. Pokud se objeví hodnota nevalidní, aplikace ji ignoruje a dále se s ní nepracuje (tzn., hodnota se neodesílá skriptu ke zpracování). Tím se zamezí několika špatným měřením, které by nepříznivě ovlivňovaly data získané z webové stránky. Takovéto nevalidní hodnoty my mohly působit jako nechtěné extrémy ve výsledném datovém souboru.

Tabulka 2.1 Popis vybraných dat přicházejících ze zařízení pro Eye Tracking

Název atributu	***Hodnoty atributu***	***Popis atributu***
TIME	*Desetinné číslo*	*Doba od inicializace nebo od poslední kalibrace zařízení*
BPOGX	*Desetinné číslo od 0 do 1*	*Horizontální souřadnice nejlepšího z očí v danou chvíli. Hodnota je zlomek z velikosti obrazovky.*
BPOGY	*Desetinné číslo od 0 do 1*	*Vertikální souřadnice nejlepšího z očí v danou chvíli. Hodnota je zlomek z velikosti obrazovky.*
BPOGV	*1 nebo 0*	*Značka určující validitu dat (1 pro validní a 0 pro nevalidní).*
CX	*Desetinné číslo od 0 do 1*	*Horizontální souřadnice nejlepšího z očí v danou chvíli. Hodnota je zlomek z velikosti obrazovky.*
CY	*Desetinné číslo od 0 do 1*	*Vertikální souřadnice nejlepšího z očí v danou chvíli. Hodnota je zlomek z velikosti obrazovky.*
CS	*0*	*Stav kurzoru myši (0 - připraven, 1 - levé tlačítko myši stisknuto, 2 - pravé tlačítko myši stisknuto). V současnosti ještě není tato funkcionalita zařízením podporována (není využíváno).*

Příklad jednoho záznamu přicházejícího ze zařízení:

```
<REC TIME="0" BPOGX="0,09635" BPOGY="0,77407" BPOGV="1"
CX="0,59635" CY="0" CS="0" />
```

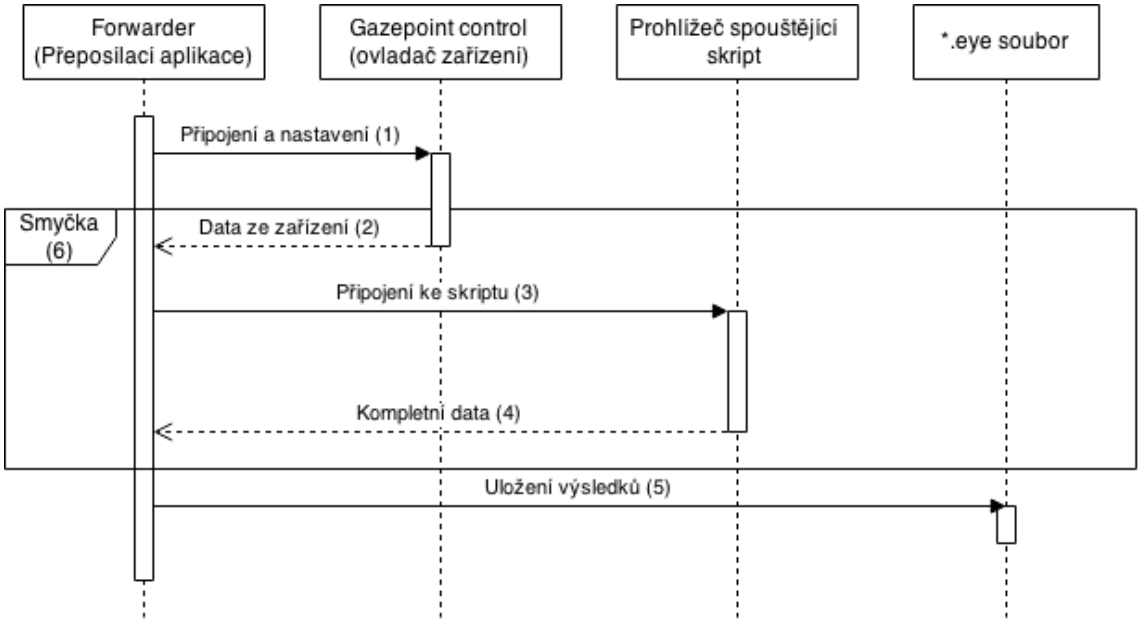
Cílem je s pomocí získaných souřadnic najít na webové stránce ten správný HTML element, na který se uživatel v konkrétní moment dívá. Krom elementu aplikace získá i doplňující informace, jako jsou pozice elementu a jeho identifikátor. Ukládají se také informace o webové stránce, která se měří. Výsledkem měření je textový soubor, v tomto případě s koncovkou eye. Ten je dále zpracováván pomocí aplikace popsané v pozdější kapitole zabývající se získáváním výsledků.

2.3 Řešení pro měření a sběr dat

V této kapitole je popsán návrh a implementace řešení pro měření uživatelova chování na webových stránkách a následný sběr získaných dat. Kapitola je rozdělena do několika bloků, které postupně snižují úroveň abstrakce.

2.3.1 Vysoko úrovňový pohled

Použité řešení se skládá z několika od sebe oddělených částí. Společně tyto části tvoří řešení, které je schopno kombinovat data ze zařízení s daty o uživatelově chování při procházení webové stránky.



Obrázek 2.2: Abstraktní pohled na architekturu řešení a komunikaci mezi jednotlivými aplikacemi v řešení pro sběr dat

První obdélník na obrázku 2.2 reprezentuje aplikaci prostředníka (respektive přeposílací server). Tato aplikace zaopatřuje transformaci a přepočet dat, přidá k nim časovou značku (TimeStamp). Přepočet znamená převedení desetinných čísel reprezentující pozice pohledu očí a pozice myši na souřadnice na obrazovce.

Druhou aplikací je software společnosti GazePoint. Ten zprostředkovává kalibraci zařízení, jeho potřebné nastavení podle předaných parametrů a pak i posílání dat přes TCP/IP proud na IP adrese 127.0.0.1 (localhost) a portu 4242.

Třetí v pořadí je prohlížeč (respektive webová stránka), ve kterém běží skript pro propojení dat ze zařízení pro sledování pozice pohledu očí a sémantiky stránky.

Výsledek měření je následně uložen do textového souboru s koncovkou eye. Po uložení se může celý cyklus opakovat a aplikace je připravená na nové měření. Všechna data jsou uloženy jako JSON objekty. V obrázku 2.2 je tato akce zobrazena šipkou s číslem 5.

Poslední nepopsanou částí obrázku je komunikace mezi jednotlivými částmi řešení, zobrazená v čase jako je u tohoto typu diagramu běžné (od shora dolů se čas zvyšuje).

První komunikací je připojení přeposílací aplikace k software pro kontrolu zařízení GazePoint a odeslání potřebných nastavení (šipka označená číslem 1). Nastavení probíhá pomocí jednoduchých zpráv, které jsou popsány dokumentaci zařízení. První dva příkazy slouží ke zjištění rozlišení obrazovky a rozlišení kamery (pouze pro informativní účely - jsou vypsány v přeposílací aplikaci). Další tři příkazy nastaví, které data má zařízení posílat. Jde o čas, nejlepší pozici pohledu očí a pozici kurzoru myši. Čtení pozice kurzoru myši až v přeposílací aplikaci mělo negativní vliv na synchronizaci a v ojedinělých případech nebyly data korektní. Proto jsem využil možnost číst tato data také pomocí zařízení. Poslední příkaz spustí samotné odesílání dat a aplikace je tak schopna číst nastavené data. Proud dat ze zařízení je v obrázku 2.2 reprezentován šipkou s číslem 2.

Ukázka nastavovacích příkazů:

```
<GET ID="SCREEN_SIZE" />
<GET ID="CAMERA_SIZE" />
<SET ID="ENABLE_SEND_TIME" STATE="1" />
<SET ID="ENABLE_SEND_POG_BEST" STATE="1" />
<SET ID="ENABLE_SEND_CURSOR" STATE="1" />
<SET ID="ENABLE_SEND_DATA" STATE="1" />
```

Pokud existuje nějaká stránka se spuštěným skriptem, je automaticky spojena s přeposílací aplikací pomocí technologie web sockets (na obrátku 2.2 zobrazeno šipkou s číslem 3). Tato technologie funguje jako upgrade protokolu http, ale s využitím protokolu TCP/IP. Ještě se ji budu více věnovat v kapitole o implementaci, ale v podstatě jde o to, že je umožněno oboustranné spojení (takzvaný full duplex) v prohlížeči. Je tak možné data posílat na webovou stránku a zároveň z webové stránky zpět do přeposílací aplikace. Aplikace tedy po připojení přeposílá data ve formátu JSON skriptu který s nimi dále pracuje. Hlavička je poslána pouze jednou při zavedení spojení (tj. při handshake) a tím pádem nevzniká tak velká datová zátěž jako při využití zpráv technologie AJAX.

Skript postupně přiřazuje elementy a jiné informace k datům, které obdrží. Postupně však kompletní data odesílá zpět na server, aby nedošlo ke zbytečné ztrátě dat, nebo k příliš obsáhlým zprávám (v obrázku 2.2 reprezentováno šipkou s číslem 4). Tento přístup se ukázal být vhodnější než odesílat nasbíraná data najednou v jedné dlouhé zprávě, převážně kvůli větší spolehlivosti tohoto postupu. Data se posílají v relativně krátkých intervalech a tak by skript mohl zabírat velké množství paměti, nebo dokonce zapříčinit nefunkčnost celé stránky. Takové chování je rizikové a raději se mu proto v aplikaci vyhýbám již dříve zmíněným odesíláním dat průběžně. Data lze odeslat i na příkaz, to znamená po obdržení konkrétní zprávy z přeposílací aplikace. Data jsou hromadně odeslána v případě zavření okna prohlížeče (resp. webové stránky) nebo pokud uživatel stiskne ukončovací sekvenci kláves. Konkrétně jde o kombinaci kláves Ctrl+Shift+S. Nemělo by tak dojít k žádné ztrátě naměřených dat (pouze při kritických chybách jako je pád prohlížeče apod.).

Příklad jednoho záznamu ve výsledném souboru eye (po odeslání z prohlížeče zpět přeposílací aplikaci):


```
{"ticks":1427449751240,"x":1342,"y":396,"mouseX":1301,"mouseY":4
06,"element":"DIV!class.row","EX":364.5,"EY":188,"EWidth":1170,"
EHeight":608,"duration":16,"dom":"HTML;BODY#page-
top;HEADER;DIV;DIV;DIV","DWidth":1899,"DHeight":3415,"WWidth":19
20,"WHeight":985,"url":"http://localhost:1646/Pages/testScenario
1.cshtml#ws://127.0.0.1:8181/"}
```

Tabulka 2.2 Popis jednotlivých vlastností shromážděných na výstupu aplikace

Jméno vlastnosti	Hodnota vlastnosti
ticks	Časová značka ve formátu pro javascript (tj. počet milisekund od 1. 1. 1970)
x	Horizontální souřadnice pohledu očí
y	Vertikální souřadnice pohledu očí
mouseX	Horizontální souřadnice pozice kurzoru myši
mouseY	Vertikální souřadnice pozice kurzoru myši
element	Nalezený element v daný okamžik ve zkráceném zápisu
EX	Horizontální pozice elementu na stránce v daný okamžik
EY	Vertikální pozice elementu na stránce v daný okamžik
EWidth	Šířka elementu na stránce v daný okamžik
EHeight	Výška elementu na stránce v daný okamžik
duration	Doba, po kterou byl element sledován (v milisekundách)
dom	Zkrácený zápis pro zaznamenání pozice elementu v DOM
DWidth	Šířka dokumentu v daný okamžik
DHeight	Výška dokumentu v daný okamžik
WWidth	Šířka okna prohlížeče (nebo záložky) v daný okamžik
WHeight	Výška okna prohlížeče (nebo záložky) v daný okamžik
url	Kompletní (tzn. včetně hash a query) URL adresa stránky měření v okamžik měření

2.3.2 Přeposílací aplikace

V této kapitole je detailněji popsán návrh a implementace přeposílací aplikace. Kapitola také obsahuje popis uživatelského rozhraní a popis některých použitých algoritmů a technik.

Mezi hlavní úkoly přeposílací aplikace patří:

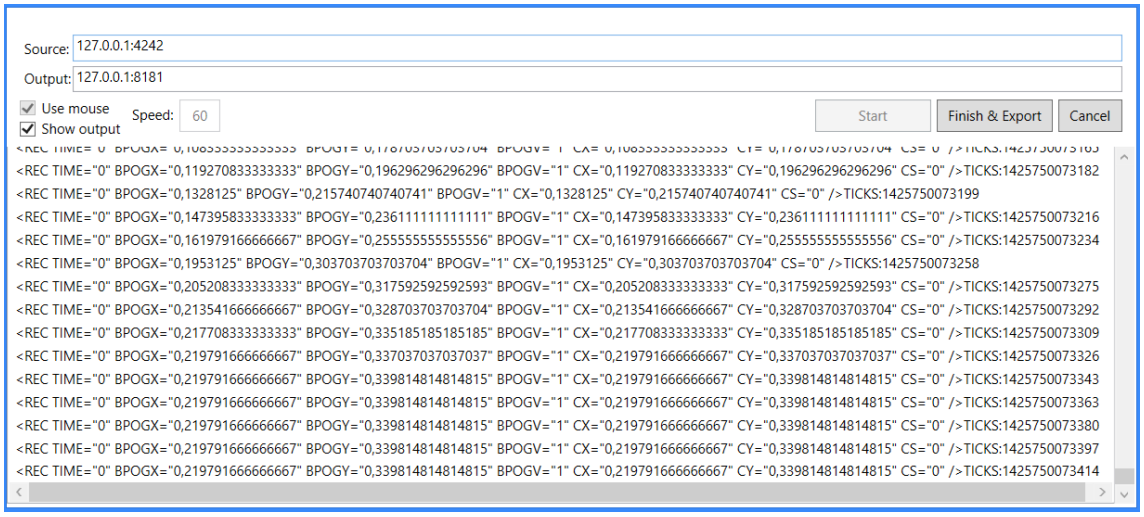
- Navázat spojení se software zařízení GazePoint
- Nastavit zařízení GazePoint
- Číst data ze zařízení GazePoint
- Navázat spojení se skriptem v prohlížeči
- Odesílat validní data skriptu k dalšímu zpracovaní
- Informovat o odesílaných datech
- Zkompletovat výsledné data a uložit je do souboru

2.3.2.1 *Uživatelské rozhraní aplikace*

Pro implementaci přeposílací aplikace je použita technologie WPF (Windows presentation foundation). Jednou z výhod WPF technologie je rychlé překreslování uživatelského rozhraní, což je nutné pro rychlý výpis přeposílaných dat. Aplikace vypisuje originální data docházející ze zařízení pro sledování pohledu očí a adekvátní časovou značku. Výpis zachovává posledních 300 záznamů. Počet je omezen kvůli zbytečné paměťové náročnosti a především přehlednosti pro uživatele. Aplikace by neměla mít problém případně zobrazovat až několik tisíc záznamů, ale takové nastavení nemá příliš význam.

Kvůli plynulosti aplikace a hlavně skriptu v prohlížeči aplikace nepřeposílá všechna data ze zařízení. Kdyby tomu tak bylo, hrozilo by zamrzávání skriptu, je potřeba vzít v potaz, že na stránce musí být schopny běžet i jiné skripty a není tak rozumné blokovat hlavní vlákno webové stránky po celou dobu měření. Z toho důvodu je v aplikaci nastaveno malé zpoždění. Zpoždění lze chápat jako funkcionalitu, která po odeslání záznamu na specifikovanou dobu ignoruje další záznamy. Nejmenší nastavitelné zpoždění je 1ms. Při tomto nastavení se může stát, že se aplikace zablokuje a nebude možné z ní exportovat data. Je proto potřebné před nastavením rychlosti aplikace uvažovat nad rychlostí hostitelského počítače. I 20 měření za vteřinu dokáže dodat dostatek dat pro správnou funkci aplikace, ale pro větší detailnost dat je možné počet dat zvýšit. Pokud je rychlost aplikace příliš malá, data nemusí být dostatečně vypovídající pro některé případy užití, ale to záleží na konkrétním měřeném scénáři.

Nevýhodou použití technologie WPF je možnost běhu pouze na systémech kompatibilních s platformou Microsoft.NET (v době vzniku této práce jde převážně o operační systémy Windows). Aplikace byla vyvinuta a testována na OS Windows 8.1 64b a použitá verze frameworku .NET byla 4.5.



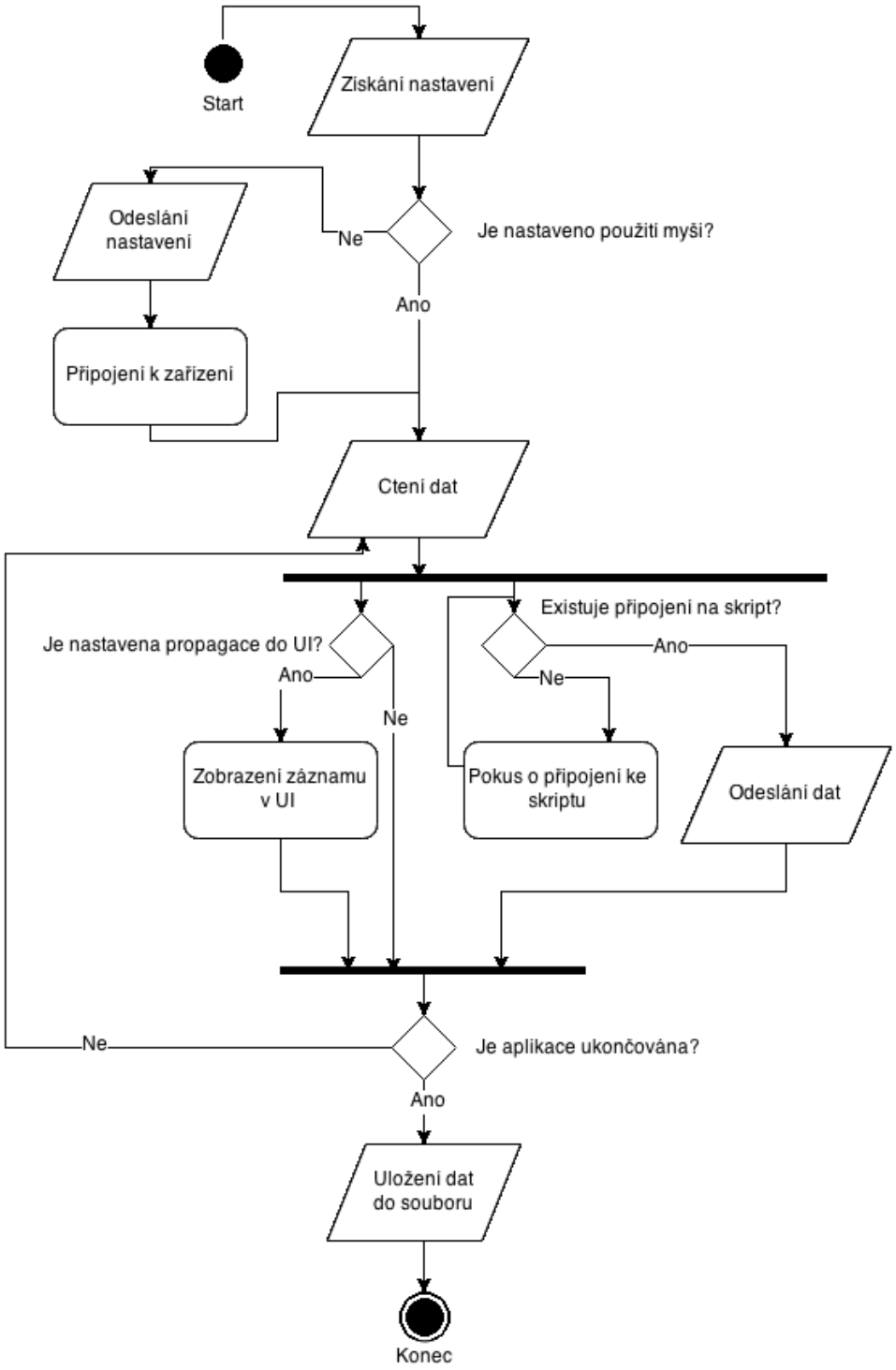
Obrázek 2.3: Ukázka uživatelského rozhraní přeposílací aplikace

Na obrázku 2.3 je zobrazeno uživatelské rozhraní přeposílací aplikace za jejího běhu. Z pole source lze zjistit, že by aplikace používala jako zdroj dat adresu 127.0.0.1:4242. To je adresa, na které posílá použité zařízení svá změřená data při původním nastavení. Díky

zatrženému nastavení "use mouse" aplikace simuluje použití zařízení pro sledování očí pomocí myši. Hodnoty pro pozici myši a zařízení jsou tak v takovém případě totožné. Výstup je nastavený na 127.0.0.1:8181. Na tomto portu je aplikace připravena obsluhovat požadavky z webových stránek a přeposílat a přijímat data. Rychlost (Speed) na obrázku je nastavena na hodnotu 60, to znamená přibližnou rychlost 60 přeposlaných měření za vteřinu (konkrétní hodnoty záleží i na rychlosti používaného hardware). Jde o nastavení, které zaručí dostatek dat a zároveň plynulost aplikace. V případě výkonnějších počítačů jde samozřejmě využít vetší rychlost měření. Posledním nezmíněným nastavením je pole "show output". Tato hodnota určuje, zda se bude zobrazovat výpis doručených dat v místě k tomu určeném (ve spodní časti uživatelského rozhraní, jak je viditelné na obrázku 2.3). Zbývá popsat tři tlačítka jakožto hlavní ovládací prvky aplikace. První s popisem "Start" spustí hlavní scénář aplikace s vybraným nastavením. Druhým tlačítkem s popisem "Finish & Export" se ukončí zaznamenávání dat a je umožněno exportování získaných dat do souboru s koncovkou eye. Třetím a posledním tlačítkem je tlačítko pro zrušení běžícího zaznamenávání s popisem "Cancel".

Hodnoty pro zdroj dat a výstup není potřeba nastavovat, pokud jsou stejné jako na obrázku 2.3 - jde o standartní hodnoty, které jsou použity v případě prázdných polí.

2.3.2.2 ***Hlavní algoritmus aplikace***



Obrázek 2.4: Abstrakce algoritmu hlavní části aplikace

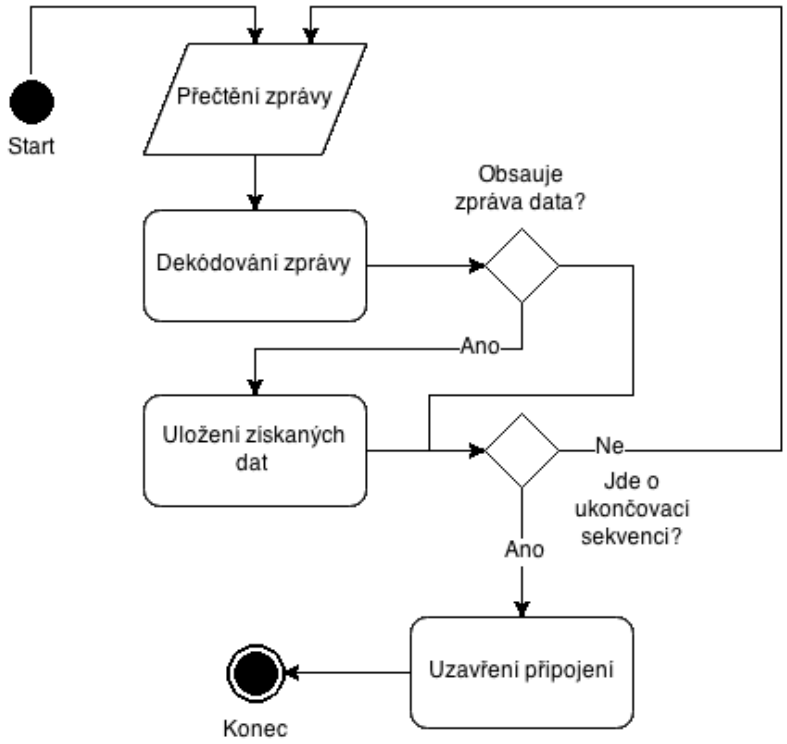
Na předcházejícím diagramu (Obrázek 2.4) je znázorněn značně zjednodušený algoritmus aplikace. Pro přehlednost v něm nejsou zobrazeny některé detaily nastavování, zpracovávání chyb a podobné části aplikace. Tento diagram slouží především jako abstrakce. Kromě popsané části, aplikace obsahuje také část přijímající data zpět ze skriptu na webové stránce. Ta bude popsána zvlášť později v této kapitole.

Aplikace nejprve získá nastavení z uživatelského nastavení. Pokud některé nastavení chybí, použijí se standartní hodnoty (jde převážně o zdroj a cíl), které směrují na adresu localhost (127.0.0.1) a porty 8181 a 4242. Pokud jsou zadány hodnoty, které nejsou převeditelné na adresy, aplikace nepokračuje. Aplikace má možnost simulovat zařízení pro snímání očí pomocí myši (převážně pro testovací účely). Jestliže není tato možnost nastavena, aplikace pokračuje odesláním nastavení do zařízení. Po obdržení nastavení ovladač zařízení začne vysílat data jako stream na protokolu TCP. V této době už je aplikace číst data nezávisle na zdroji. Ke každému záznamu aplikace připojí ještě aktuální čas v adekvátním formátu, aby bylo následně možné provést simulaci. Při každé doručené zprávě se data propagují na potřebná místa. Volitelně jde o zobrazování dat v uživatelském rozhraní a vždy jde o odesílání do skriptu (samozřejmě jen pokud je nějaký k aplikaci připojen).

Než se data odešlou, kontroluje se, zda existuje připojení na skript ve webovém prohlížeči. Pokud ještě žádný skript není připojen (např. jestliže ještě stránka nebyla zobrazena, uživatel přešel na novou stránku apod.) znovu se vytvoří připojení pomocí třídy Socket, která asynchronně čeká na připojení ze strany skriptu. Ke konkrétní implementaci připojení se ještě vrátím v další části této kapitoly. Pokud aplikace není ukončována, celý proces se opakuje, dokud existují data ke čtení.

Ukončení aplikace může být spuštěno buď ze strany skriptu (uživatel odeslal ukončovací sekvenci) nebo ze strany aplikace samotné (uživatel ukončil své měření a chce uložit výsledek). Ukončování je reprezentováno slovem "completed" na začátku zprávy. Posledním krokem je jen uložení sesbíraných dat do souboru s koncovkou eye. Než se data uloží, aplikace pošle zprávu "collect". Tato zpráva informuje skript o tom, že má odeslat zbývající data a ukončit spojení. Odpovědí na zprávu "collect" jsou data společně se zprávou "completed". Po tomto ukončení jsou potřebná spojení uzavřena a aplikace je schopna uložit data do souboru s koncovkou eye.

2.3.2.3 *Pohled na komunikaci se skriptem v prohlížeči*



Obrázek 2.4: Znázornění algoritmu pro příjem dat ze skriptu

Obrázek 2.4 poskytuje detailnější pohled na přijímání a zpracovávání zpráv ze skriptu. Po obdržení zprávy je potřeba ji dekódovat, protože protokol WebSockets automaticky všechny zprávy kóduje. Způsob kódování je shodný s tím v dokumentu RFC 6455 a jeho implementace bude detailněji popsána v kapitole o protokolu WebSockets. Po získání čitelné podoby dat je řetězec validován a tím je zjištěno, jestli obsahuje data v potřebném formátu. Pokud tomu tak je tak se záznam vloží do kolekce, která je k tomu určená. Tato kolekce obsahuje všechny získané data od doby spuštění aplikace až do uložení dat do výstupu (soubor eye). Než aplikace pokračuje k další zprávě, zkontroluje, zda jde o ukončovací sekvenci. Ukončovací sekvence znamená, že řetězec začíná slovem "completed". Pokud jde o ukončovací sekvenci, aplikace ukončí spojení se skriptem ve webovém prohlížeči.

Aplikaci je možné rozdělit do dvou hlavních paralelních částí. První zaopatřuje odesílání dat ze zařízení pro sledování pohledu očí ve formátu JSON, který je připojený k aplikaci a druhá naslouchá zprávám ze skriptu. Zprávy ze skriptu obsahují data obohacené o informace o stránce a hlavně konkrétním sledovaném elementu. Odesílání zpráv je také použito ve speciálním případě ukončování aplikace. Zamezí se tak situaci, ve které by mohlo dojít ke ztrátě získaných dat, protože se data odesílají po blocích pěti záznamů (v originálním nastavení, které je možné změnit, velký počet není doporučený). V opačném případě by tak mohla nastat situace, při které se až 4 záznamy ztratí. Samozřejmě, že v ojedinělých případech jako je pád prohlížeče nebo jiná kritická chyba může nastat ztráta dat, ale i v takových případech jako jsou tyto, je vždy většina dat získána. Data se ukládají v aplikaci pro přeposílání a ne v prohlížeči.

Tím je umožněn záznam z několika propojených stránek do jedné společné kolekce. Je tedy potřeba brát v potaz tuto funkčnost, aby nedošlo k situaci zaznamenávání několika nesouvisejících případů užití do jedné kolekce. Data by v takovém případě nedávaly příliš smysl a jen těžko by byly použitelné. Nicméně je možné, že specifické zadání umožní i takovou funkčnost využít. Mohlo by jít například o porovnání chování na více podobných stránkách najednou. Porovnání zvlášť se ale zdá jako příjemnější a srozumitelnější možnost.

2.3.2.4 ***Implementace protokolu websockets***

Jak již zmíněno dříve, pro komunikaci mezi skriptem v prohlížeči a aplikací pro přeposílání dat je využita technologie zvaná WebSockets. Pomocí jiných technologií, jako je například technologie AJAX, by takováto funkčnost byla proveditelná, ale znamenala by mnohem méně efektivní komunikaci. Pro prohlížeče, jenž nepodporují technologii WebSockets se často využívá takzvaného long polling. Při této strategii klient odesílá standartní asynchronní požadavek na server a ten odkládá odeslání odpovědi. Tím je simulováno obousměrné spojení. Znamená to ale značně zvýšenou zátěž jak na síti, tak na serveru. Klient navíc může čekat dlouho na neukončený požadavek.

Pro tuto aplikaci se předpokládá využití moderních prohlížečů s podporou protokolu WebSockets. WebSockets protokol je standardizován IETF a popsán v dokumentu RFC 6455, ze kterého má implementace vychází. Prohlížeče poskytují API pro tento protokol, a proto není potřeba implementovat klientskou stranu komunikace. Protokol je založený na TCP a využívá protokol HTTP pro nastavení a zavedení komunikace. Pro zavedení komunikace je nutné použít definovaný handshake. Handshake využívá podpory Upgrade pole v hlavičce HTTP, což je funkcionalita tohoto protokolu od verze 1.1.

Příklad pokusu o navázání komunikace ze strany prohlížeče (s využitím předem definovaného API prohlížeče Google Chrome 40, implementace se může mírně lišit u jiných prohlížečů nebo verzí):

```
GET http://127.0.0.1:8181/ HTTP/1.1
Host: 127.0.0.1:8181
Connection: Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket
Origin: http://localhost:1646
Sec-WebSocket-Version: 13
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.115
Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: cs-CZ,cs;q=0.8
Sec-WebSocket-Key: nFejc+xUw4/YncSWV7BbhA==
Sec-WebSocket-Extensions: permessage-deflate;
client_max_window_bits
```

Na tomto příkladu je zřetelné, že pro nastavení komunikace se využívá metody GET protokolu HTTP ve verzi 1.1. Hodnota connection oznamuje změnu protokolu a v poli upgrade je pak nastavována informace přímo o použití WebSocket protokolu. Verze protokolu je v tomto případě verze 13. Odesílaná hlavička handshake obsahuje i další informace. Jde například o pole user-agent, které předává informaci o použitém prohlížeči (v tomto případě jde o dříve zmíněný prohlížeč Chrome ve verzi 40.0.2214.115). Od verze prohlížeče se může odvíjet verze protokolu WebSockets (podle verze, kterou implementuje konkrétní prohlížeč respektive verze prohlížeče). Hodnota accept-encoding a accept-language informují druhou stranu připojení o přijímaném kódování a o používaném jazyce. Jednou z povinných a důležitých hodnot je Sec-WebSocket-Key. V tomto poli je uložena hodnota klíče pro odpověď ve formátu base64 řetězce. Server tak klientovi zaručí, že jde o validní otevírací handshake protokolu WebSocket. Jde také o bezpečnostní výhodu, která znemožňuje cizím ajaxovým požadavkům vydávat se za zprávu protokolu WebSocket. Další specifikum pro protokol websockets je hodnota Sec-WebSocket-Extensions, která předává druhé straně informaci o použitých rozšíření protokolu. V tomto případě jde o rozšíření permessage-deflate a client_max_window_bits. Permessage-deflate oznamuje použití takzvaného DEFLATE algoritmu pro kompresi dat. Algoritmus DEFLATE je popsán v RFC 1951 [12] a použitá metoda byte boundary aligning v dokumentu RFC 1979.

Dalším použitým rozšířením je client_max_window_bits. To pomáhá snížit množství využívané paměti při přenosu dat.

Odpověď na tento požadavek může vypadat takto:

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Accept: CLVBmuG8gnY3WbMVydxmUcJEpuI=
```

Jako odpověď je použita ta nejkratší možná. Stejně jako požadavek využívá také protokolu HTTP ve verzi 1.1, ale jako status kód se odesílá hodnota 101 signalizující změnu protokolu (Switching Protocols). Nejdůležitější hodnotou, kromě informativních hodnot Connection a Upgrade, které slouží k podobným účelům jako v případě požadavku, je hodnota Sec-WebSocket-Accept. Jde opět o hodnotu zakódovanou ve formátu base 64 řetězce, která vznikne následujícím algoritmem kde "requestHeaderKey" reprezentuje hodnotu pole Sec-WebSocket-Key z požadavku skriptu a hodnota guid je GUID definovaný ve specifikaci protokolu WebSocket (konkrétně jde o "258EAFA5-E914-47DA-95CA-C5AB0DC85B11") .

```
Convert.ToBase64String(
 SHA1.Create().ComputeHash(
  Encoding.UTF8.GetBytes(
   requestHeaderKey + guid
)))
```

Nejdříve se tedy hodnota převede na pole bajtů v kódování UTF8, po té se vytvoří hash pomocí algoritmu SHA1 a výsledek se převede na base64 řetězec. Po odeslání této hlavičky se provede připojení a obě strany jsou schopny číst i posílat data.

Pro síťovou komunikaci na straně přeposílací aplikace je využito tříd typu Socket. Tyto třídy jsou součástí .NET frameworku od verze 4.0. Díky těmto třídám je umožněna asynchronní práce s komunikací na síti, kód tedy není zbytečně blokující. Veškerá implementace týkající se této části aplikace je ve třídě ForwarderLib.Server.

2.3.2.5 ***Souborová struktura aplikace***

Celý projekt pro přeposílací aplikaci je rozdělen do tří knihoven. Forwarder (tj. WPF aplikace obsahující převážně práci s uživatelským rozhraním aplikace), ForwarderLib (tj. knihovna obsahující převážně logiku aplikace) a ForwarderTests (tj. místo pro testy aplikace).

Seznam a použití souborů v řešeních:

- Forwarder
    - MainWindow.xaml - soubor obsahující implementaci uživatelského rozhraní hlavního okna aplikace)

    - MainWindow.xaml.cs - soubor obsahuje logiku interakce s uživatelským rozhraní
- ForwarderLib
    - Client.cs - obsahuje logiku týkající se komunikace, odesílání a čtení dat ze zařízení pro snímání pozice očí na obrazovce
    - Data.cs - takzvaný ViewModel pro hlavní okno aplikace
    - FileHelper.cs - soubor zastřešující práci se soubory (např. pro ukládání výsledného souboru)
    - IpWorker.cs - pomocná třída pro práci s IP adresami a jejich převodem
    - Server.cs - obsahuje vše co se týká komunikace se skriptem v prohlížeči, včetně práce se získanými daty
    - Utils.cs - ostatní jinde nezařazená funkčnost (např. logování chyb do souboru)

2.3.3 Zaznamenávací skript

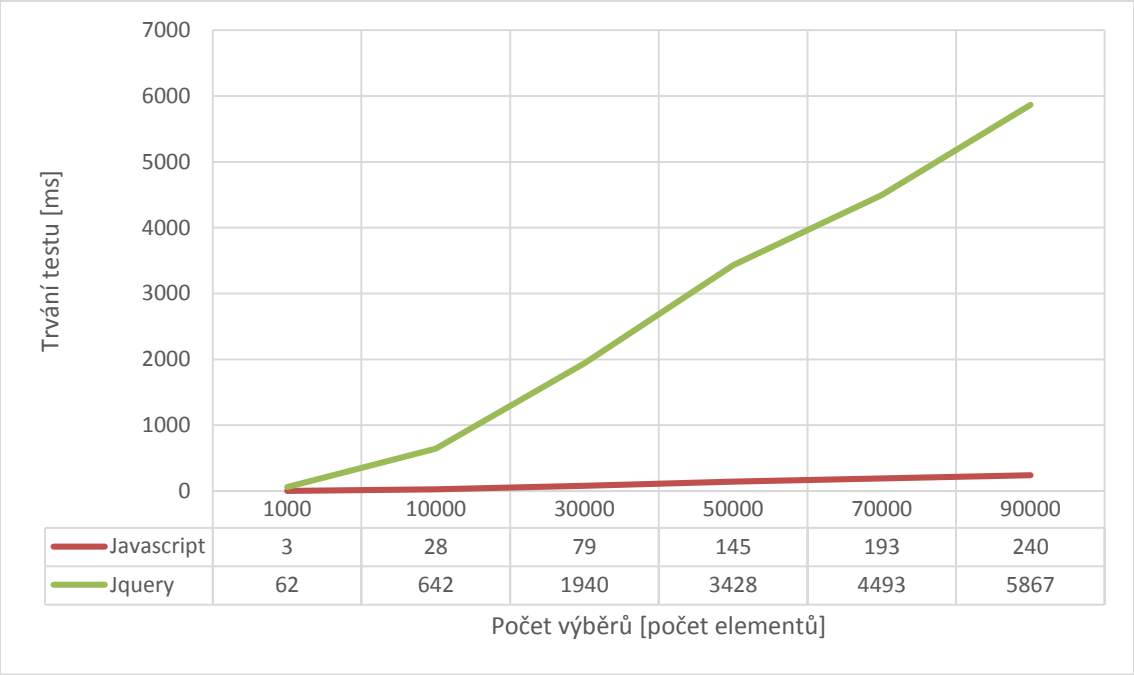
V této kapitole je popisován skript, který představuje další podstatnou část mého řešení. Je to část, která se připojí k přeposílací aplikaci (ta je popsána v předešlé kapitole) a s využitím dat získaných od eye tracking zařízení zjistí další informace pro analýzu.

Hlavní úkoly skriptu lze rozepsat takto:

- Připojit se k přeposílací aplikaci
- Přijímat data z přeposílací aplikace
- Identifikovat konkrétní HTML element ležící na pozici pohledu
- Získat informace o nalezeném HTML elementu a jeho pozici ve struktuře DOM
- Zformátovat data do potřebného formátu
- V intervalech odesílat data zpět přeposílací aplikaci

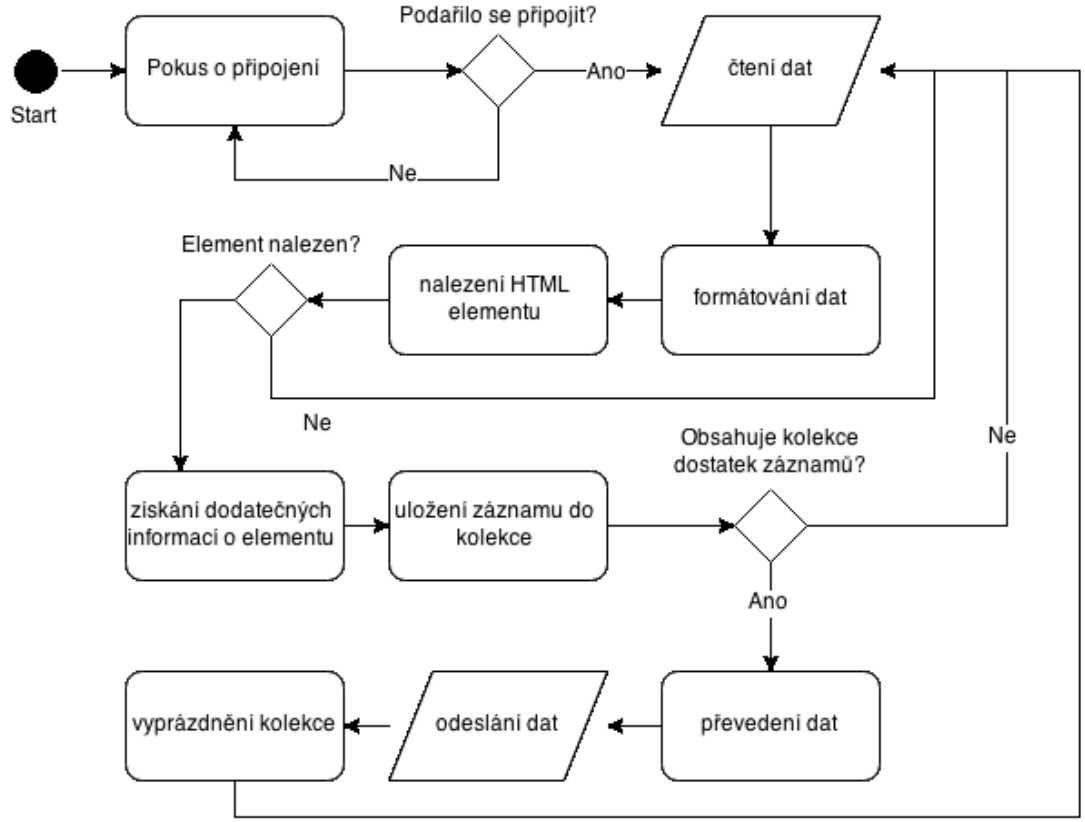
2.3.3.1 Návrh skriptu

Často je bráno jako samozřejmost, že ruku v ruce s aplikací v jazyce Javascript jde i použití četných frameworků a knihoven, jenž mají usnadnit práci s tím to jazykem. Pro psaní skriptu jsem se ovšem rozhodl nevyužít žádnou z nich. Jako jeden hlavní důvod považuju rychlost aplikace a především práci s DOM jako je zobrazeno na následující grafech. Skript musí být schopen obsloužit až desítky požadavků za vteřinu a rychlost práce s DOM je častým důvodem pro pomalý běh javascriptových aplikací.



Obrázek 2.5: Grafické porovnání rychlostí výběru HTML elementů u čistého Javascriptu a knihovny jQuery

Z obrázku 2.5 je zřetelné, že použití knihovny jQuery sice ulehčuje práci s DOM, ale zpomalení je znatelné. Protože má aplikace provádí velké množství podobných požadavků rozhodl jsem se použít čistý Javascript (tzn. bez frameworků a jiných doplňků). Ani při použití knihoven by zpomalení nebylo takové, aby znemožnilo plynulé fungování aplikace. Test byl proveden na stránkách http://www.zive.cz/ dne 15. 12. 2013 v prohlížeči Opera Next 19.0.1326.9 (v ostatních prohlížečích jsou hodnoty podobné).



Obrázek 2.6: *Zjednodušený náhled na algoritmus skriptu*

Na obrázku 2.6 výše je znázorněn diagram algoritmu skriptu. Diagram je určený jako abstrakce a nejsou v něm zobrazeny veškeré detaily algoritmu ani všechny detaily implementace.

2.3.3.2 ***Popis algoritmu skriptu***

Po načtení webové stránky se skript pokouší v pravidelných intervalech o délce 2 sekund připojit k přeposílací aplikaci. Adresu aplikace skript primárně získává z URL adresy, kde může být zanesena jako hash část adresy (příklad #ws://127.0.0.1:8181). Pokud není hash část adresy ve správném formátu k dispozici, skript jako druhou možnost hledá adresu v cookie s klíčem "ws". Pokud je adresa nastavena, automaticky se uloží do příslušné cookie a tak není nutné ji přeposílat na další související měřené stránky.

Po úspěšném připojení je zaregistrována událost naslouchající zprávám. Událost onmessage je na obrázku 2.6 reprezentována jako čtení dat. Po přečtení dat je nutné hodnoty převést z globálních souřadnic na souřadnice lokální. Globální souřadnice lze v tomto kontextu chápat jako pozici od prvního pixelu obrazovky. Pozice [0,0] by tedy byla prvním pixelem levého horního rohu. Lokální souřadnice jsou ty, které používá jazyk Javascript. Souřadnice takové, kde pozice [0,0] reprezentuje první pixel webové stránky (resp. První pixel horního levého rohu webové stránky). Dalším faktorem ovlivňujícím pozice na obrazovce jsou hodnoty zarolování webové stránky. Některé funkce jazyka javascript totiž považují první pixel horního

levého rohu za pozici [0,0]. Je tedy nutné počítat se zarolováním webové stránky, velikostí okna webové stránky ale i velikostí uživatelského rozhraní. Hodnoty se mohou měnit i v průběhu měření, proto je nutné neustále přepočítávat (tzn. při každém novém záznamu).

Nalezení HTML elementu znamená identifikaci HTML elementu, který leží pod danou pozicí pohledu očí. Tento úkol by znamenal časté a zdlouhavé postupné procházení struktury dokumentu (resp. DOM), nebýt funkce document.elementFromPoint. Tato funkce vrací strukturálně nejvýše položený dokument ležící pod předanými souřadnicemi. Pokud je některý z argumentů této funkce negativní, nebo jeho velikost přesahuje velikost viewport (lze chápat jako oblast zobrazení, jde o hodnotu CSS) v daném rozměru, funkce vrací null. Pokud funkce nenajde žádný konkrétní element a souřadnice jsou správné velikosti, vrátí kořenový element dokumentu. Je důležité zmínit, že funkce ignoruje elementy, na nichž je nastavena CSS vlastnost pointer-events na hodnotu none. Aby se toto chování obešlo, skript nastaví hodnotu této CSS vlastnosti na "pointer-events: all !important;" u nejvýše položeného elementu struktury dokumentu (tj. HTML). Tím se tato hodnota propaguje na všechny elementy v dokumentu. Ovšem toto chování nemusí být žádoucí u všech webových stránek, je nutno brát v potaz to, že vypnutí pointer-events může mít u některých stránek specifický důvod a tyto elementy by se neměly měřit. Toto chování je tedy doporučené zvážit případ od případu (tzn. pro každou webovou stránku alternativně). Případně lze tuto vlastnost obejit nastavením parametru important i u elementů, jenž mají být skriptem ignorovány.

V této fázi již je nalezen element. Dříve, než skript pokročí dál, zkontroluje, zda nejde o element, který by měl být považován za nezobrazený nebo neviditelný. Pokud tomu tak je, hledá se rekurzivně takový rodič, který může být považován za viditelný element. Tento element pak nahradí ten neviditelný.

Dalším krokem je kontrola zda nejde o totožné měření, jako v posledním případě. Pokud ano, pouze se zvýší vlastnost duration u předchozího elementu. Duration se v přeposílací aplikaci násobí rychlostí, takže nedojde k nesymetrii ani ztrátě dat. Za výhodu to má mírné zkrácení algoritmu. Totožným měření rozumím ekvivalentní element a ekvivalentní hodnoty pozice myši a pozice očí. Pro kontrolu ekvivalence elementů jsou využity metody isEqualNode a isSameNode. První z nich kontroluje, zda elementy mají totožné atributy, hodnoty atributů, jména a jsou stejného typu. Druhá metoda kontroluje, zda jde o stejný element. Obě metody jsou potřeba z důvodu časté změny elementů při prohlížení stránky (Javascriptová manipulace s elementy, hodnoty polí, změna třídy apod.).

Posledním krokem před získáním všech dalších informací o elementu je kontrola, zda se nejedná a fixně pozicovaný element. Skript opravuje pouze pozice fixed a ne pozice sticky, protože implementace chování při nastavení pozice na sticky není příliš rozšířená (v době implementace ji podporují pouze prohlížeče s minimální verzí Firefox verze 35, Safari verze 7.1 a iOS Safari verze 7.1) a chování není ekvivalentní ani u existujících implementací. Nicméně pokud se v budoucnu používání tohoto nastavení rozšíří, bude nutné i tuto možnost zvážit a případně doimplementovat podle vybrané specifikace prohlížeče. Opravou pozice je myšleno její správné umístění v dokumentu. Například pokud je stránka zarolovaná o 100 pixelů a

element s pozicí fixed je na pozici [0,0] jeho pozice se bude jevit jako [0,100]. To znamená, že by ve výsledných datech byl tento element na mnoha místech (při každé změně zarolování stránky by měl nové umístění) a znečistil by výslednou simulaci. Element se může vyskytovat na více místech, pokud je animován nebo jinak manipulován (například manipulace v Javascriptu), ale pokud jde o fixní element je pro simulaci výhodnější, když se jeho zarolovaná pozice ignoruje. Pro jinak animované elementy je nová pozice důležitá z hlediska designu, proto se při záznamu nijak neupravuje.

Nalezený element je v této době připravený pro získání ostatních informací. Ty se zapíší ve formě Javascriptového objektu do připravené kolekce k dalším měřením. Při získávání informací o elementu se bere v potaz pozice okna prohlížeče, velikost okna prohlížeče, zarolování po obou osách i velikost dokumentu. Aby bylo možné později rekonstruovat pozici elementu v DOM, musí být zapsána jeho cesta k HTML elementu. Ačkoli existuje funkce řešící tento problém (document.elementsFromPoint), její implementace není spolehlivá a použití není příliš časté. Pokud se implementace této metody v budoucnu natolik zlepší, že bude ověřeně používaná, mohla by nahradit mou použitou implementaci.

Pro zjištění cesty ke kořenovému prvku dokumentu tedy používám vlastní rekurzivní funkcí. Tento přístup má za výhodu dostatečnou rychlost a možnost zapsat elementy tak, jak se to pro mou implementaci nejvíce hodí. Kromě jména elementu jsou zapsány ještě informace o jeho identifikátoru (pokud existuje). Jména elementů jsou odděleny znakem ";" a hodnoty identifikátorů znakem "#" (tzn. zachování syntaxe css selektorů). Ostatní atributy u předchozích elementů nejsou zaznamenávány, aby se předešlo příliš dlouhým řetězcům. Kvůli relativně častému měření je výhodnější pracovat s menšími objemy dat a pro výslednou rekonstrukci tento zkrácený způsob záznamu nemá příliš negativní dopad.

Před ukončením záznamu se provádí kontrola množství prvků v kolekci s měřeními. Pokud její velikost přesahuje maximální nastavenou (v původním nastavení je to 5 elementů), prvky se odešlou přeposílací aplikaci ke zpracování a uložení. Kolekce je po té vymazána, nikdy tedy nemá víc než stanovený počet prvků. Nemělo by tedy docházet ke zpomalení prohlížeče z důvodu nedostatku paměti. Vyprázdnění a odeslání kolekce s daty nastane i při přečtení zprávy "collect". Po odeslání odpovědi na tuto zprávu se připojení ukončuje.

2.3.4 Obecné informace o používání

V této kapitole je popsáno, jakým způsobem lze aplikaci použít, aby nedošlo k problémům nebo nedorozuměním. Je také popsán očekávaný případ užití a předpokládané chování aplikace.

Součástí kompletního řešení a tedy i hlavního scénáře užití jsou čtyři aplikace.

- Software pro práci se zařízením pro snímání pohledu očí GazePoint (tzn. aplikace Gazepoint Control)
- Přeposílací aplikace (tzn. Forwarder)
- Skript na měřené webové stránce (tzn. Analyser.js)

- Aplikace pro usnadnění analýzy (tzn. AnalyserW8)

Pro hostování webových stránek je taktéž přiloženo řešení s několika testovacími stránkami. Toto řešení ale není nutností a skript může bez problémů běžet samostatně na jakékoli jiné webové stránce, proto tato aplikace není uvedena v předcházejícím seznamu. Aplikace pro usnadnění analýzy není přímou součástí sběru dat, ale může fungovat samostatně, pokud má všechny potřebné vstupy (tj. eye soubor s daty a případně snímek webové stránky).

Na pořadí mezi přeposílací aplikací a skriptem na webové stránce nezáleží. Pokud přeposílací aplikace nevyužívá simulaci pomocí myši, je nutné, aby první běžel ovladač zařízení pro snímání pozice očí. Bez ovladače zařízení nic nevysílá a není tak možné přeposílat data.

Popis očekávaného scénáře použití pro měření:

1. Uživatel spustí software zařízení pro sledování pozice očí Gazepoint control
2. Uživatel provede kalibraci zařízení (pomocí software zařízení)
3. Uživatel spustí přeposílací aplikaci Forwarder
4. Uživatel nastaví přeposílací aplikaci
5. Uživatel zapne přeposílací aplikaci
6. Uživatel vidí ve výstupu přeposílací aplikace informace o přicházejících datech
7. Uživatel v libovolném prohlížeči podporujícím technologii websockets zobrazí stránku obsahující skript Analyser.js
8. Uživatel provede svůj test na webové stránce
9. Uživatel ukončí snímání (opuštění stránky, kombinace kláves Ctrl+Shift+S nebo ukončení v přeposílací aplikaci)
10. Uživatel uloží výsledný soubor s daty

2.3.5 Problémy se získáváním dat z webových stránek

Webové stránky a aplikace jsou jedním z nejvíce pestrých druhů software a mohou se lišit v mnoha faktorech. Jedním z takových faktorů je způsob implementace průchodu stránkami. Webové aplikace jde rozdělit na stránky klasické, kde každá stránka představuje samostatný požadavek na server, jednostránkové aplikace kde se co nejvíce dat získává asynchronně s využitím AJAXových požadavků (případně s využitím podobných technologií) nebo na aplikace využívající kombinaci těchto přístupů. Ačkoli mnou implementované řešení je schopno pracovat na několika na sebe navazujících stránkách (klasické webové aplikace), výhodnější je využití na jednostránkových aplikacích (SPA) u kterých většina zdrojového kódu a tím pádem i sémantika webu zůstává stejná. Rekonstrukce takových stránek je pro uživatele přehlednější a jednodušší. Při vícestránkových aplikacích je nutné později data filtrovat tak, aby adekvátně fungovala pro cíle měření.

Kvůli této rozmanitosti webových stránek je složité vytvořit řešení, které by fungovalo ve všech případech podle očekávání. Během implementace popisovaného řešení jsem narazil na několik takových speciálních případů v chování některých elementů. Šlo například o problém s vlastností pointer-events popsanou dříve nebo o problémy se specifickým umístěním elementů, viditelností, dynamickou změnou velikosti uživatelského rozhraní a další. I když se mi objevené

problémy podařilo vyřešit, nebo je obejít jiným řešením, můžou stále existovat takové aplikace (nebo webové stránky), kde mé řešení naruší buď funkčnost aplikace, nebo naopak aplikace naruší funkci skriptu. V těchto situacích jde o kompromisy a konkrétní řešení můžou vyžadovat pozornost při nastavení některých vlastností aplikace.

Již dříve jsem zmínil, že aplikace očekává použití některého z moderních prohlížečů. Nejvíce z mého testování proběhlo v prohlížečích Google Chrome 40 a Opera 28. Prohlížeče na většině mobilních zařízeních a na operačních systémech jiných než Microsoft Windows testovány příliš nebyly a je u nich možné mírně odlišné chování. Základní funkčnost by měla zůstat stejná i v nadcházejících verzích prohlížečů a operačních systémů.

3 Rozbor a analýza změřených dat

3.1 Popis problému

Výstupem zatím popsaných aplikací je soubor obsahující data zaznamenávající vlastnosti proběhlých měření. Problémem je, že v této fázi jde pouze o data nikoli o informace. Data tedy nejsou pro čtenáře nijak zajímavá a je proto nutné s nimi dále pracovat, tak aby z nich dokázal snadno přečíst potřebné informace. Informace lze z dat získat jejich zpracováním a analýzou. K hledání informací v datech lze využít techniky takzvaného dolování dat (Data mining, DM).

Při dolování dat lze využít velké množství metod a postupů včetně algoritmů umělé inteligence a statistiky. K dolování dat se také váže několik procesů, které se snaží poskytnout návod (resp. postup) pro získání co nejlepší šance na nalezení hledaných informací. Z mnoha procesů týkající se dané problematiky zde zmíním například proces SEMMA, který se příliš nezabývá aspekty byznysu, nebo později vyvinutý proces CRISP-DM, který se skládá z 6 částí (včetně části pochopení byznysu). [13]

Tabulka 3.1 Porovnání vybraných procesů pro dolování dat z hlediska jejich fázi

SEMMA proces	CRISP-DM	KDD proces
Sample (vybírání dat)	Business understanding (pochopení cílů, plánování)	Selection
Explore (prozkoumání)	Data understanding (pochopení dat)	Pre-processing
Modify (upravení)	Data preparation (příprava dat)	Transformation
Model (aplikace modelování)	Modeling (aplikace modelování)	Data Mining
Assess (posouzení)	Evaluation (vyhodnocení)	Interpretation/Evaluation
	Deployment (prezentace dat)	

Z tabulky 3.1 je viditelné porovnání vybraných procesů. Krom několika sémantických rozdílů v pojmenování jednotlivých částí se procesy jeví jako podobné. Vždy je data nejdříve nutné získat, následně se s daty seznámit, připravit data pro danou metodu (algoritmus atd.), následně výsledky vyhodnotit a posoudit jejich správnost. Mírně rozdílný se jeví proces CRISP-DM. Zakončení u něj není vyhodnocením dat, ale až jejich prezentací.

Každý algoritmus pro dolování dat je vhodný pro určitý typ dat, nebo pro získávání určitého druhu informací. Zatím neexistuje spolehlivá metoda schopna zaručit získání všech potřebných informací z libovolných dat v libovolném formátu. Cílem takovýchto postupů a algoritmů je převést data do srozumitelné a reprezentovatelné podoby, která je snadno pochopitelná pro toho kdo informace v datech hledá.

3.2 Řešení problému

Dalším logickým krokem v honbě za informacemi je vytvoření aplikace, která by napomohla a podporovala získávání informací z vytvořeného datového souboru. Při postupování dle procesu CRISP-DM je potřeba splnit následující kroky:

- Business understanding (Plánování)
    - Tato fáze se zaměřuje na pochopení cílů projektu a požadavků z perspektivy byznysu a následné převedení těchto informací na definici problému dolování dat.
- Data understanding (Pochopení dat)
    - Fáze porozumění dat se snaží o pochopení dat. Cílem je detekovat možné problémy s daty a hledat zajímavé části dat, případně informace, které nejsou na první pohled zřetelné.
- Data preparation (Příprava dat)
    - Fáze přípravy dat spočívá v akcích, které vedou vytvoření datového souboru připraveného pro další analýzu. K tomu se využívá dat z původního datového souboru.
- Modeling (Aplikace modelování)
    - V této fázi jsou vybrány postupy a algoritmy pro hledání informací, které se pokouší vyřešit definované problémy. Tyto algoritmy jsou aplikovány na připravený datový soubor.
- Evaluation (Vyhodnocení)
    - Před pokračováním do finální fáze je důležité ověřit si, že provedené postupy a algoritmy splnily to, co se od nich očekávalo. Na konci této fáze je učiněno rozhodnutí o tom, jak použít zatím získané informace.
- Deployment (Prezentace dat)
    - Poslední fáze spočívá v prezentaci informací získaných v předešlých krocích tak, aby byly srozumitelné a jednoduše čitelné.

V některých případech je nutné se k některým fázím opakovaně vracet po přezkoumání v další části. Například z modelování do přípravy dat, nebo z vyhodnocení do plánování. [13]

Cílem je zjistit, zda monitorování pozice pohledu očí společně se sémantikou webových stránek funguje a jestli tento způsob analýzy webu dokáže poskytnout nějaké cenné informace. Výsledkem analýzy by měla být rozhodnuta otázka, zda má toto řešení potenciál při reálném využití v praxi, případně o jaké využití by se mohlo jednat. Díky výsledné aplikaci by uživatel měl mít možnost sledovat chování měřených uživatelů na webu, zjistit jaké elementy uživatelé sledovali nejčastěji, na které elementy se jejich zrak nejčastěji vracel a jaký je vztah pohledu očí a pozice myši na obrazovce.

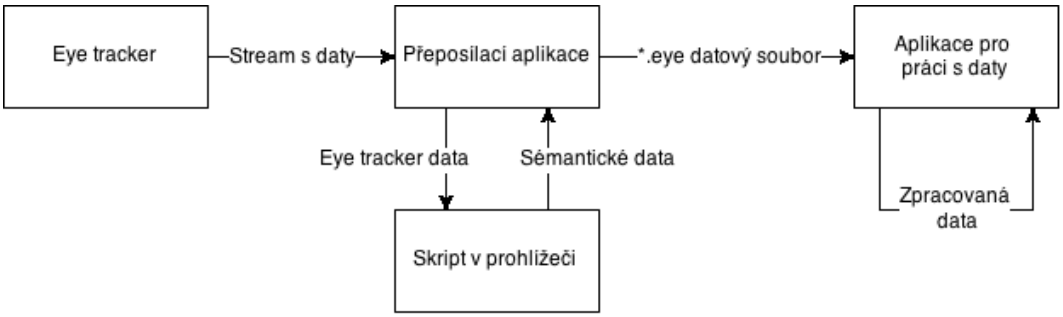
Výhodou je, že data jsou z velké části již ve vhodném formátu a proto jejich pochopení a příprava nebude náročná. Některá příprava byla již provedena v předchozích krocích při

vytváření datového souboru. Použité algoritmy pro přípravu dat a jejich modelování jsou popsány v následující kapitole o aplikaci pro práci s daty.

Výsledné informace by měly být uživateli zobrazeny ve srozumitelné formě, která nabízí snadnou prezentaci získaných výsledků. Protože dat může být z celého měření hodně, aplikace musí být schopna zobrazit jak část, tak všechna data. Kdyby byla aplikace schopna zobrazit pouze celý datový soubor najednou, mnoho informací by bylo pro uživatele ztraceno díky příliš komplikovaným výstupům.

3.3 Aplikace pro práci s daty

V této kapitole je popsána architektura a design aplikace, která má za úkol prezentovat data v takové podobě, aby informace z nich byly snadno čitelné a srozumitelné. Aplikace je výsledkem snahy splnit očekávání a požadavky nastolené při návrhu řešení problému (popsáno v kapitole 3.2).

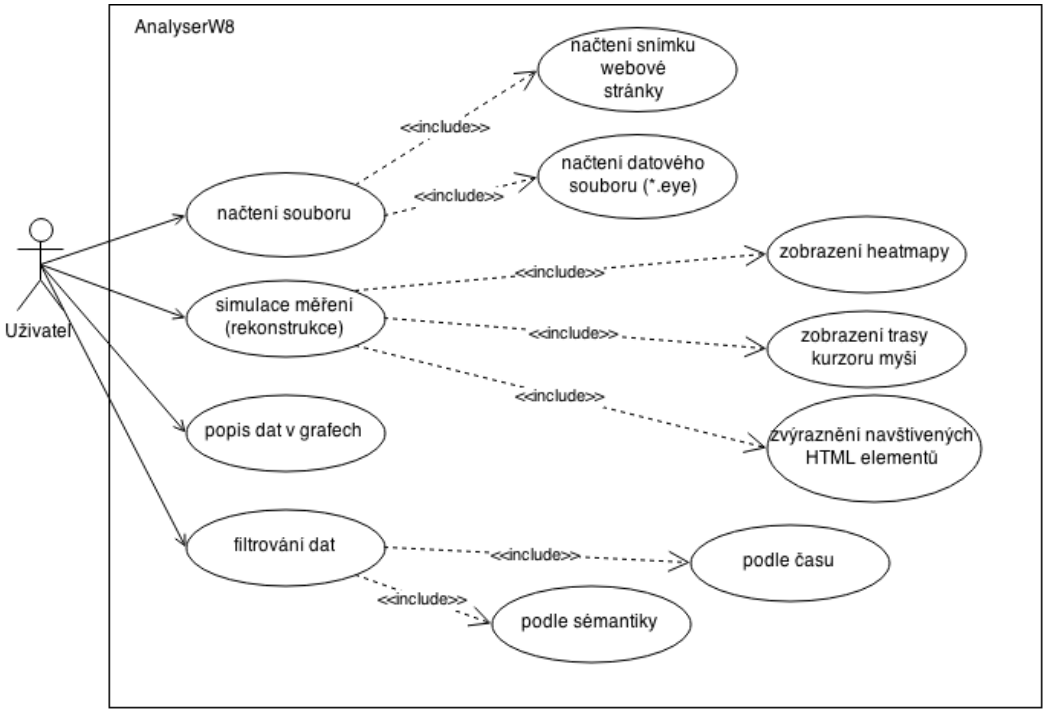


Obrázek 3.1: Pohled na celkovou architekturu řešení. Výsledkem jsou zpracovaná data, nad kterými je možné provést potřebnou analýzu.

3.3.1 Návrh a funkce aplikace

Protože velkou částí aplikace je její vizuální stránka, bylo použito prostředí Microsoft Windows 8. Toto prostředí poskytuje celoobrazovkové prostředí pro aplikace s moderním designem. Tento přístup je ideální pro aplikace zobrazující velký počet dat. Uživatelské rozhraní aplikace nezabírá téměř žádné místo ve vykreslovací ploše.

Pro aplikace v prostředí Windows 8 existují dvě možnosti implementace. První z nich je využít programovacího jazyka C# pro logiku aplikace a jazyk XAML pro její stránky (views). Druhou možností je využít implementace programové logiky v jazyce Javascript a stránky pomocí jazyka HTML. Protože podstatnou částí aplikace je její vzhled a pro programovací jazyk Javascript s HTML existuje velké množství knihoven a frameworků zaměřující se na tuto problematiku, rozhodl jsem se využít této možnosti implementace.



Obrázek 3.2: Diagram případu užití pro aplikaci pracující s daty

Na obrázku 3.2 je znázorněn diagram případu užití pro aplikaci, která má za úkol pomoci získat informace o datech (tj. AnalyserW8). V diagramu je popsána funkcionalita implementované aplikace.

Aplikace je schopna načítat dva typy souborů. Jedním z nich je datový soubor s koncovkou eye vytvořený pomocí dříve popsané přeposílací aplikace. Tento soubor musí obsahovat validní data ve formátu JSON. Volitelně může aplikace načíst obrázek webové stránky (to může být například tzv. screenshot), aby výsledná simulace respektive rekonstrukce byla lépe zasazena do kontextu.

Další podstatnou funkcí aplikace je možnost zrekonstruovat měření. V této rekonstrukci je zobrazen jak pohled očí, tak pozice myši uživatele. Na pozadí jsou zároveň viditelné obrysy HTML elementů, které v danou chvíli ležely pod pozicí pohledu očí. Protože absolutně přesná pozice očí není díky možným nepřesnostem zařízení pro snímání pohledu jistá, aplikace používá k reprezentaci tohoto parametru takzvanou heatmapu. Heatmapa je schopná reprezentovat dobu pohledu na danou oblast barvou. Teplejší barvy (barvy s vyšší vlnovou délkou) znamenají vyšší výskyt pohledu očí na tuto pozici a studenější barvy (barvy s nižší vlnovou délkou) znamenají řidší výskyt. Využití heatmapy je také výhodou, protože velikost lidského ostrého vidění může mít velikost několik milimetrů (až centimetrů) a ne jenom bodu (pixelu) [1].

Aby uživatel dostal dobrou představu o datech, se kterými pracuje v simulaci (resp. rekonstrukci), jsou mu tytéž data nabídnuta ve formě několika grafů. Díky těmto grafům je pro uživatele jednodušší mít konkrétní představu o navštívených elementech a době, po kterou uživatel daný typ elementu sledoval. Dále je uživateli zobrazena reprezentace frekvence

sledování konkrétního typu elementu. Kromě elementů je uživateli zobrazeno i porovnání pozice myši a pozice pohledu v daném časovém úseku, nebo na vybraných HTML elementech. Tyto informace slouží mimo jiné pro zjištění uživatelova pohybu po stránce.

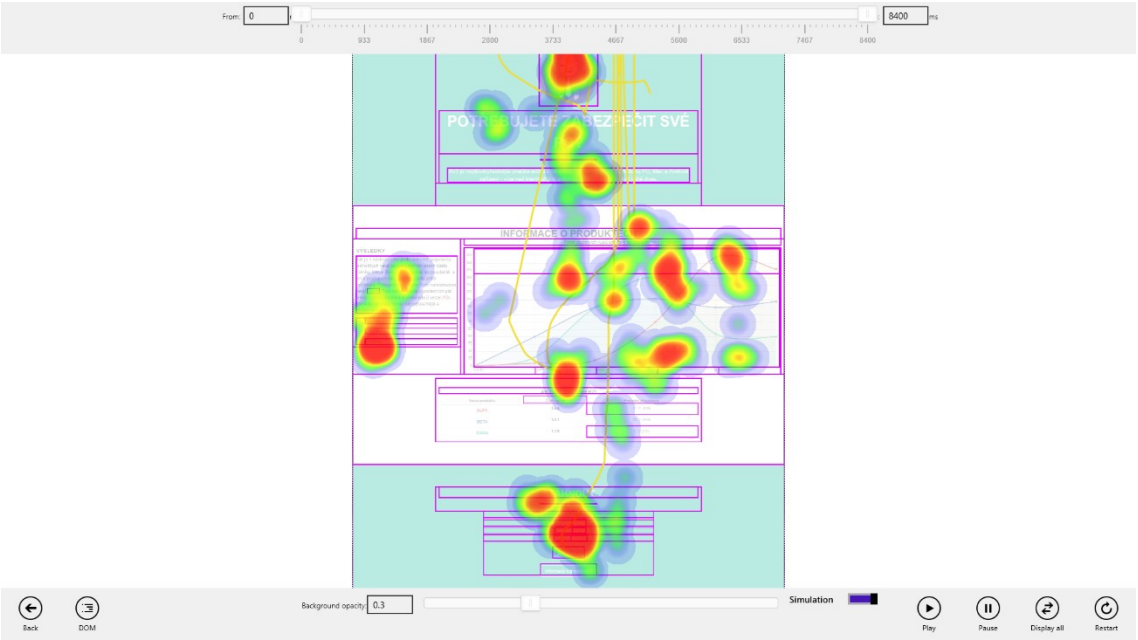
Aplikace umožnuje dva různé způsoby filtrování dat, pro co nejjednodušší manipulaci s daty.

Prvním ze způsobů filtrování je filtrování podle času. Díky tomuto filtrování je uživateli umožněno například zobrazit posledních 10 vteřin zobrazovaného měření. Díky tomuto filtru lze také porovnávat různé měření mezi sebou a to, i pokud mají různou délku trvání.

Druhým způsobem filtrování je filtrování podle vybrané sémantiky. Aplikace zrekonstruuje strukturu měřeného HTML elementu a uživatel může libovolně vybrat elementy a sekce, které jsou předmětem jeho zájmu v danou chvíli. Protože vybraný element se v měření může vyskytnout na začátku i na konci, nejde filtrování podle času a podle elementů kombinovat.

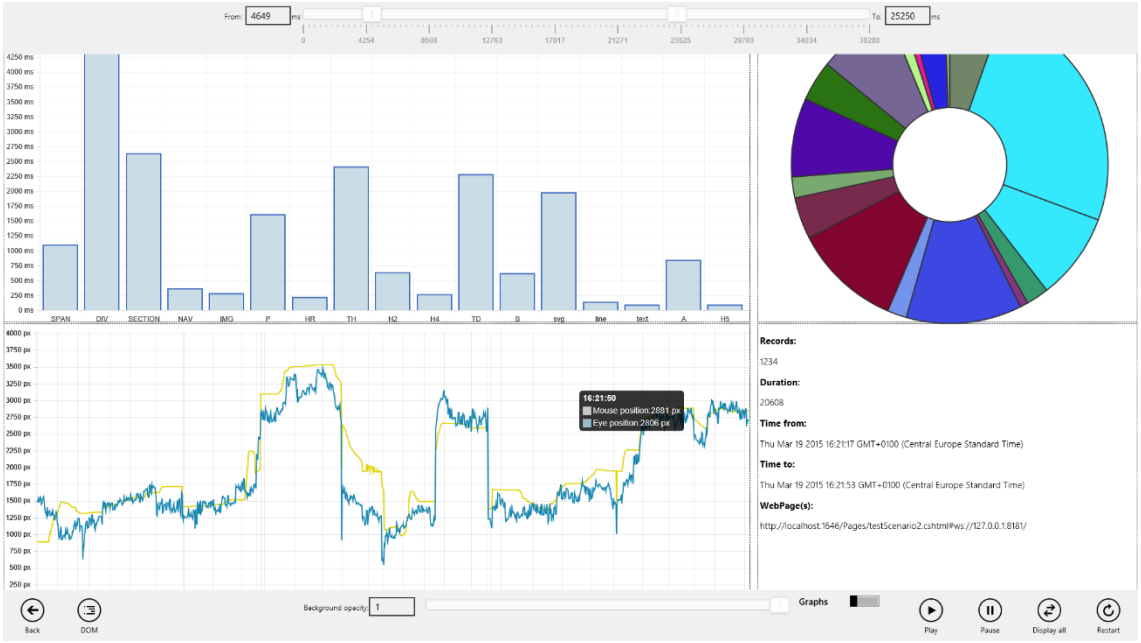
3.3.2 Pohled na použité uživatelské rozhraní

Uživatelské rozhraní aplikace je rozděleno do dvou hlavních částí. První část slouží pro načtení vstupů do aplikace. Z této části není možné pokračovat, pokud nejsou nahrány všechny potřebné data ve správných formátech. Druhá část obsahuje ostatní části aplikace vykreslující rekonstrukci měření a zobrazující další detaily o datech.



Obrázek 3.3: První stránka hlavní časti aplikace. Ve středové části je největší možná (tzn. vypočítaná podle rozměrů největší webových stránek z datového souboru v poměru k velikosti obrazovky) vykreslovací plocha pro danou webovou stránku. Vykreslovací plocha obsahuje heatmapu reprezentující pohledy měřeného subjektu, elementy sledované v průběhu měření

(fialové obdélníky) a pozici kurzoru myši (béžová čára). Na pozadí je zobrazen snímek měřené stránky s průhledností 70%. Možnost zobrazení snímku stránky je volitelná.



Obrázek 3.4: Druhá stránka zobrazená po přepnutí pohledu ze simulace na grafy. Stránka je rozdělená na 4 hlavní sekce. První sekce obsahuje graf shrnující dobu strávenou pohledem na daný typ HTML elementu. Druhá část obsahuje graf zobrazující frekvenci pohledu na konkrétní elementy (kolikrát se uživatel k elementu svým pohledem navracel). Třetí graf zobrazuje vzdálenost mezi kurzorem myši a pozicí pohledu očí. Hodnota je úhlopříčkou mezi šířkou a výškou v daném bodě. Poslední sekce zobrazuje souhrnné informace o zobrazovaných datech. Velikost jednotlivých sekcí je vypočítána tak, aby zůstala čitelná i při změně rozlišení nebo orientace obrazovky. Zobrazení grafů lze tedy měnit otočením obrazovky, pokud tuto funkci hostitelské zařízení podporuje. Některé prvky rozhraní jsou viditelné až po interakci s uživatelem a nejsou na tomto obrázku dobře viditelné. Jde například o popisy jednotlivých sekcí, či legendy pro jednotlivé grafy.

3.3.3 Konkrétnější informace o implementaci

Jak již bylo odůvodněno v části, pro implementaci jsem zvolil platformu Microsoft Windows 8 aplikace s využitím Javascriptu jako hlavního programovacího jazyku. Pro implementaci těchto aplikací je k dispozici knihovna WinJs. Díky využití implementace v Javascriptu je možná případná transformace aplikace na webovou stránku. V takovém případě by bylo nutné provést jen několik málo změn (např. změnit použití lokálního úložiště). V této kapitole je detailněji popsaná implementace aplikace pro práci s daty.

3.3.3.1 Použité knihovny a nástroje

Knihovna WinJs je určena pro vývoj v prostředí Windows 8 a nabízí nástroje pro urychlení vývoje a zlepšení jeho kvality. Knihovna je publikována pod licencí Apache 2.0 a je

součástí projektu Microsoft Open Technologies. WinJs poskytuje nástroje pro vytvoření prvků infrastruktury, jako jsou třídy, jmenné prostory a další. Knihovna také nabízí mnoho ovládacích prvků a zapouzdřené funkcionality. Ovládací prvky implementované knihovnou mají totožný vzhled jako ostatní Windows 8 aplikace. Není tak na první pohled možné poznat, jestli se jedná o aplikaci implementovanou pomocí jazyka C# a XAML nebo jazyka Javascript a HTML.

Kromě WinJs je v řešení použito několik dalších knihoven, které urychlují implementaci responsivních, funkčních a estetických komponent.

Jednou z nich je knihovna Kinetic.js usnadňující manipulaci s hlavním plátnem (canvas) pro vykreslování struktury stránky a pozice ukazatele myši při daném měření. Výhodou této knihovny je možnost rozdělení plátna na vrstvy. U vrstev je pak možné přepínat viditelnost, průhlednost, nebo je mazat. Plátno v řešení je rozděleno na 3 vrstvy. První je vrstva pro obrázek pozadí (snímek webové stránky), druhá vrstva je použita pro vykreslování pozice a velikosti HTML elementů a třetí slouží pro vykreslování pozice kurzoru myši v čase.

Další knihovna je použita pro vykreslování samotné heatmapy (heatmap.js). Tato knihovna umožnuje vytvoření a upravování heatmapy podle aktuálních hodnot. Děje se tak neustálým upravováním maximální hodnoty v zobrazení. Použitá implementace je dostatečně výkonná a stylově vyhovující vzhledu aplikace. Jako hlavní barvy heatmapy jsou použity modrá (resp. RGB[0, 0, 255]), zelená (resp. RGB[0, 255, 0]) a červená (resp. RGB[255, 0, 0]). Barva v daném bodě se vypočítá jako určitý poměr k největší hodnotě v dané chvíli. Při změně maxima se tedy musí celá heatmapa přepočítat. V případě potřeby lze díky této knihovně získat i hodnotu v konkrétním bodě. Nevýhodou je relativně málo kontroly nad vykreslovanými hodnotami.

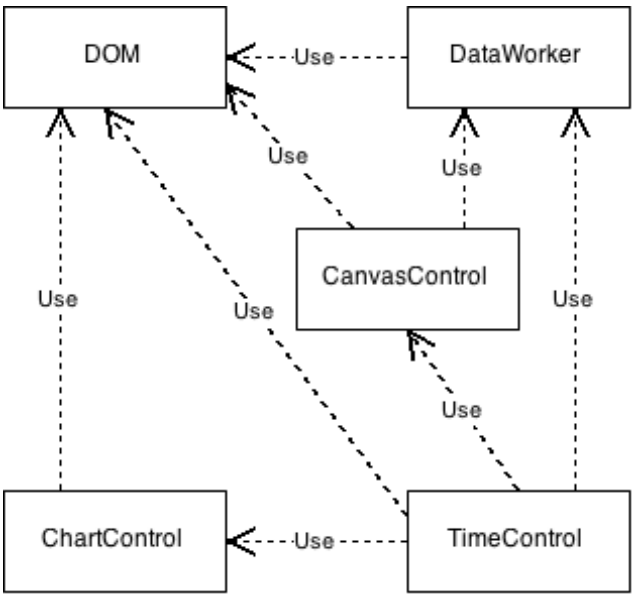
Třetí použitou knihovnou je Chart.js. Chart je použitá pro vykreslování jednotlivých grafů v části aplikace, která vizualizuje data. Knihovna je použita hlavně z estetických důvodů a z důvodů responzivního zobrazení na mobilních zařízeních a displejů různých velikostí. Z této knihovny jsou v aplikaci používány 3 druhy grafů. První je sloupcový graf pro zobrazení doby sledování jednotlivých typů HTML elementů, druhým je takzvaný "doughnut" graf (koláčový graf se středovým výřezem) a třetí použitý je graf spojnicový pro reprezentaci vzdáleností pozic kurzoru myši a pohledu očí. Pro každý z grafů jsou data transformována do adekvátního formátu, který konkrétní graf využívá.

Pro usnadnění manipulace s DOM prvky je využita i knihovna jQuery. JQuery je používána jenom v místech kde mírně menší rychlost výběru není překážkou pro správou funkčnost dané funkcionality. V místech náročných na rychlost jsou využívány nativní funkce Javascriptu.

Pro vyřešení posuvníku se dvěma ovládacími prvky je použita knihovna noUiSlider.js. Jde o rozšíření knihovny jQuery (tkz. jQuery Widget). Knihovna je použita, protože WinJs neobsahuje adekvátní implementaci podobného posuvníku. Implementovaný posuvník je použitý pro ovládání časové osy a nastavování průhlednosti pozadí (klasický s jedním ovládacím prvkem). Na události posuvníků jsou navázány konkrétní překreslovací funkce.

3.3.3.2 Vnitřní aplikační struktura

Implementace obsahuje několik WinJs tříd (nejde o Javascript třídy-použitá verze jazyka třídy nepodporuje). Třídy jsou rozdělené podle jejich funkcionality. Třída TimelineConrol zaopatřuje veškerou práci s časem a animacemi. Animace jsou převážně použity ve funkcionalitě rekonstrukce chování uživatel na webu. Pro manipulaci s daty je vytvořena třída DataWorker. Třídu DataWorker využívají všechny ostatní třídy pro přístup k potřebným datům. O samotný výběr a filtrování dat se stará třída DataWorker. Je tak zaručen jediný bod přístupu k datům. Poslední dvě ještě nezmíněné třídy jsou CanvasControl a ChartControl. První zmíněná obsahuje funkcionalitu pro kreslící vlákno použité pro funkce rekonstrukce pohybu uživatele po webu včetně heatmap, vykreslování HTML elementů a pozice kurzoru myši (první stránka hlavního okna aplikace). Druhá zmíněná třída se naopak stará o vykreslování všech grafů a souhrnných informací (druhá stránka hlavního okna aplikace). Pro funkcionalitu ohledně ovládacích prvků a nastavení, filtrování a chování aplikace je použit hlavní Javascriptový soubor příslušné stránky (main.js, default.js). V těchto souborech se také registrují události myši (resp. ovládacích prvků) a inicializují ostatní třídy.



Obrázek 3.5: Reprezentace závislostí mezi třídami a objekty v aplikaci

Obrázek 3.5 znázorňuje některé závislosti mezi objekty. Každá z tříd využívá DOM pro výběr elementů, případně pro jejich manipulaci. TimeControl je hlavní kontrolující třída, která pracuje s ostatními třídami. Například při vykreslování jednotlivých snímků, třída TimeControl získá data od třídy DataWorker a tyto data pak předá jako parametr metodám vykreslující jednotlivé částí aplikace. Ať už jde o třídu ChartControl, nebo CanvasControl jejich struktura je podobná.

Třída ChartControl obsahuje metody pro inicializaci jednotlivých grafů. Každá taková metoda nastavuje konfiguraci grafu, případně nastaví i původní data. Data jsou v případě potřeby převedena do adekvátního formátu pro konkrétní graf.

```
datasets: [{label: "Html elements",
            fillColor: "rgba(151,187,205,0.5)",
            strokeColor: "rgba(18,70,178,0.8)",
            highlightFill: "rgba(151,187,205,0.75)",
            highlightStroke: "rgba(151,187,205,1)",
            data: (function () {
                if (dataSource && dataSource.BarData) return data
Source.BarData;
                else return [];})()}
```

Na kódu výše je příklad konfigurace dat pro zobrazení v grafu (v tomto případě jde o sloupcový graf). Detailní konfigurace jednotlivých grafů je zveřejněna v oficiální dokumentaci knihovny Chart.js. Kromě inicializačních metod pro každou ze čtyř částí zobrazení třída obsahuje i metody pro aktualizaci grafů. Tyto funkce aktualizují data pro konkrétní graf a adekvátním způsobem provedou překreslení dané oblasti. Ke každé konkrétní části zobrazení se tedy vážou dvě metody. Jedna pro inicializaci a vykreslení konkrétních dat (využívané například pro vykreslení aktivity uživatele mezi druhou a čtvrtou vteřinou měření) a druhá pro aktualizaci údajů v sekci (využívané především při běhu simulační funkce). Třída obsahuje kromě těchto popsaných i několik dalších pomocných funkcí, které zde nebudu konkrétně rozepisovat, protože jejich implementace je dostatečně přímočará a mohou tak být rychle pochopeny přímo z pohledu na kód.

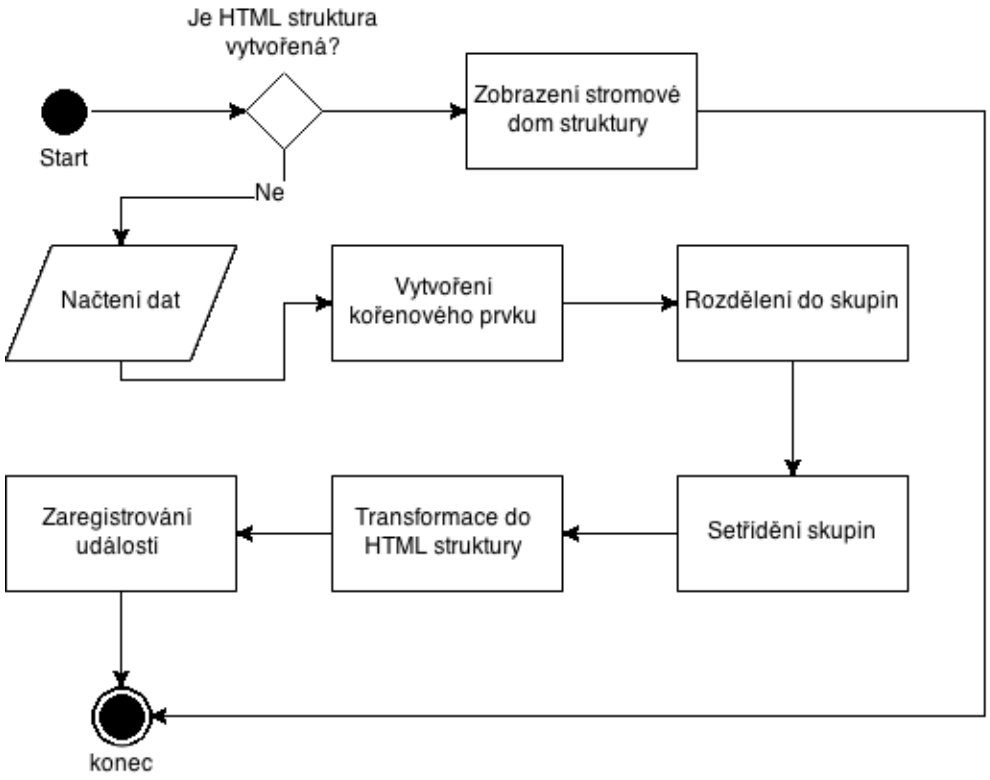
Druhou třídou zabývající se vykreslováním obsahu, je třída CanvasControl. Tato třída využívá jak knihovnu Kinetic.js, tak heatmap.js pro vykreslování sledovaných HTML elementů, pozice myši a pozice pohledu očí. Třída obsahuje, kromě pomocných a inicializačních metod, metody drawKeyFrame a drawKeyFrames. První z nich vykresluje jeden záznam a druhá záznamů několik. Opět jako u tomu je u třídy ChartControl se vykreslování jednoho záznamu využívá pro rekonstrukci chování uživatele v čase a vykreslení sekce u většiny ostatních funkcí aplikace. Každou z těchto metod lze rozdělit na další tři části. Tyto části jsou reprezentovány metodami drawElement respektive drawElements (pro vykreslení velikosti a pozice sledovaného elementu respektive elementů), drawMousePosition respektive drawMousePositions (pro vykreslení pozice kurzoru myši respektive její cesty) a drawEyePosition respektive drawEyePositions (pro vykreslení nebo aktualizaci heatmapy). Dále třída obsahuje metody pro nastavování velikosti plátna podle dat (nalezení stránky s největšími rozměry a následné přepočítání do vhodných rozměrů obrazovky) a průhlednosti pozadí (obrázek webové stránky). Všechny zakreslované hodnoty musí být tedy nejprve přepočítány na adekvátní poměr.

Nejvíce výpočetně náročnou a problémovou částí aplikace je rekonstrukce chování uživatele pomocí získaných dat. Protože měření dat může být podstatně rychlejší než vykreslování těchto dat do simulace nebo do grafů, je v některých případech nutné zvětšit

zpoždění při vykreslování. Podle mého testování jsem se rozhodl pro minimální zpoždění o velikosti 50ms. Toto zpoždění podstatně zvýší plynulost aplikace i použitelnost uživatelského rozhraní, ale na počítačích se slabším výkonem může nastat i přes toto zpoždění dočasné zamrznutí uživatelského rozhraní. Nejdůležitější metody pro tuto funkcionalitu jsou umístěny ve třídě timelineControl. Tato třída spouští vykreslování jak pro třídu CanvasControl tak pro třídu ChartControl a lze ji pro to chápat jako určitý ovládací prvek vykreslování. Pro simulaci v čase je využíváno několik metod této třídy, za zmínku stojí například metoda play. Tato metoda postupně získává data a spouští pro ně příslib (promis) v určených časových intervalech (pokud je interval větší než 50ms je použita hodnota z dat, jinak je použita hodnota 50ms). Příslib funguje podobně jako funkce setInterval v čistém javascriptu (tuto funkci není možné ve Windows 8 aplikacích využit). Po uplynutí nastaveného intervalu se postupně aktualizuje plátno simulace s heatmapou i potřebné grafy. Může se zdát zbytečné vždy aktualizovat obě stránky, ale umožní to možnost plynulého přechodu mezi tím, co chce uživatel zobrazit (ve všech případech jsou zobrazeny aktuální data). Po tomto vykreslení se volání funkce play opakuje (jde o rekurzivní funkci) a cyklus se pokračuje, dokud jsou k dispozici další data nebo dokud uživatel nestiskne tlačítko pro pozastavení, případně zrušení rekonstruování. I když je kompletní implementace této funkcionality poněkud složitější, nemá podstatný význam ji tu více do detailů rozepisovat.

Poslední důležitou nepopsanou třídou je DataWorker. Tato třída filtruje a předává konkrétní požadovaná data. Ty jsou pak využívány v místě potřeby (většinou jde o dříve zmíněnou třídu TimelineControl). Kromě metod pro filtrování a výběr dat třída obsahuje i funkci pro vypočtení velikosti okna a délky měření, protože tyto věci se také týkají dat (je tak zaručeno, že k datům je přistupováno jenom z této třídy).

Některé funkce nejsou rozděleny do konkrétních tříd. Jednou takovou je rekonstrukce a následný výběr dat podle sémantické struktury měřené webové stránky.



Obrázek 3.6: Zjednodušená reprezentace algoritmu pro vytvoření HTML stromové struktury z načtených záznamů. Pro každý soubor se převedení provádí maximálně jednou. Prvky jsou seskupeny podle jejich elementů (včetně jejich tříd a identifikátorů). Skupiny jsou následně setříděny podle jejich pozice v DOM struktuře (element HTML je kořenová skupina). Ze zápisu v řetězci se vytvoří HTML struktura pomocí seznamů tak, aby reprezentovala pozice jednotlivých elementů v původním HTML dokumentu. Před ukončením jsou zaregistrovány události pro výběr elementů a následnou propagaci do zobrazení. Je tak možné filtrovat zobrazení na základě sémantické struktury měřeného dokumentu.

3.4 Výsledky a zjištění

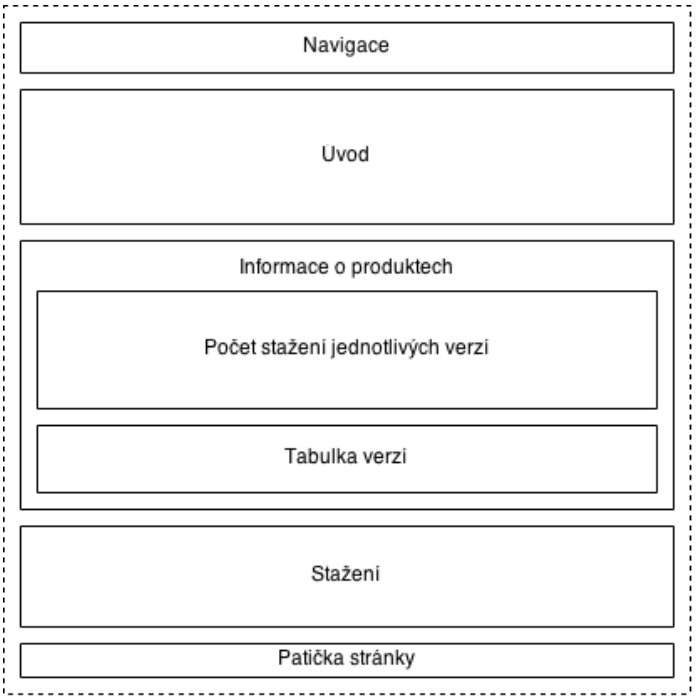
3.4.1 Popis měření

Abych zjistil, jestli je mé řešení funkční a použitelné provedl jsem několik měření na reálných (testovacích) uživatelích. Prováděné měření je možné rozdělit do dvou hlavních kategorií.

Jednou kategorií, která testuje především funkčnost řešení na různých zdrojových kódech a stylech je použití reálných stránek. Vybrané reálné stránky byly zkopírovány z internetu a upraveny tak, aby byly schopné běžet na lokálním prostředí. Toto vyžadovalo některé úpravy v kódu. Především šlo o změnu některých skriptů a zdrojů (to jsou převážně stylovací css soubory a obrázky). Změna byla nutná, protože kód a některé zdroje narušovaly pravidla stejného zdroje (Same Origin Policy). Další změnou bylo odebrání analytických a

jiných skriptů, které by teoreticky mohly nepříznivě ovlivnit fungování stránek z lokálního zdroje. Sémantika a HTML struktura jednotlivých stránek zůstala bez větších změn. Některé ze stránek nepochází se skutečného prostředí a byly vytvořeny čistě pro účel testování funkčnosti řešení u specifických HTML elementů.

Druhou a podstatnější kategorií testování je měření na dvou testovacích stránkách. Jde o stránky vytvořené přímo pro tento účel. Pro jejich design jsou použité populární jednostránkové šablony (Grayscale a Freelancer). Funkčností i stylem jsou stránky podobné, ale uspořádání informací a barevné schéma (respektive design) se mírně liší.

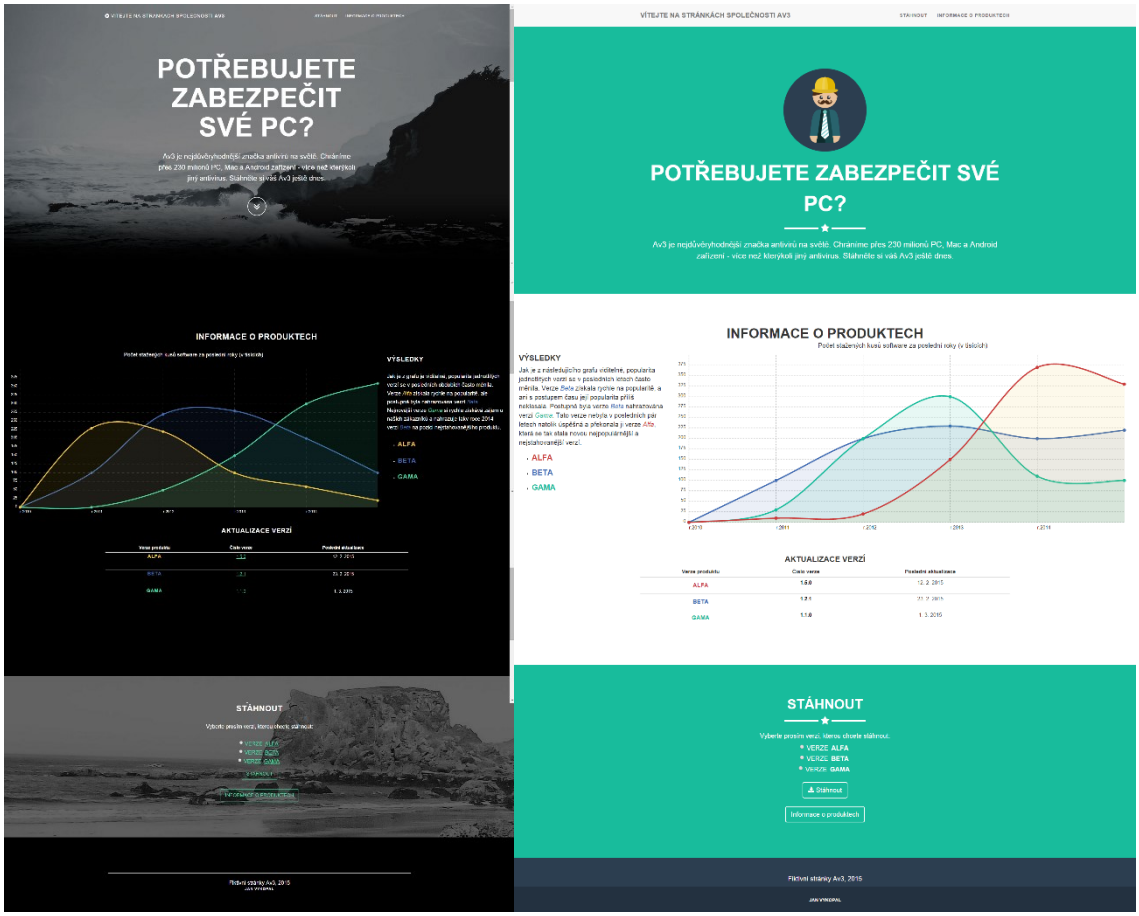


Obrázek 3.7: Rozložení testovacích stránek

3.4.2 Testovaný scénář

Před měřením dostal každý měřený uživatel pokyny sdělující informace o tom co má na další stránce provést. Obě testovací stránky byly fiktivními stránkami společnosti prodávající několik verzí produktu (pracovní názvy pro verze jsou alfa, beta a gama). Před určením vhodného scénáře jsem vytvořil několik různě složitých úkolů, různé rozložení stránek a různé množství poskytnutých informací. Stránky často obsahovaly široké spektrum sémantických prvků pro testování různých prvků rozhraní. Po několika měřeních se ukázalo, že měření uživatelé jsou často zmatení, neví, co na stránkách hledají, případně nejsou schopni dokončit zadaný úkol. I když se funkčnost řešení prokázala pozitivně, nebylo možné řešení příliš porovnávat. V případě některých testů byly požadované úkoly splněny, ale měření probíhalo nepřiměřeně dlouho. Proto jsem se rozhodl použít jednodušší scénář měření. Výsledným zjednodušováním stránek i testovacího scénáře jsem dospěl k finální verzi, kterou jsem měřil na několika různých uživatelích.

Úkolem každého testovaného uživatele bylo identifikovat a stáhnout produkt s určenými specifiky. Pro jeden scénář šlo o produkt s největším počtem stažení za rok 2014 a pro druhý testovací scénář a produkt s největším počtem stažení za rok 2013. Grafy specifikující počet stažení za konkrétní rok pro dané verze se u obou testovacích stránek liší, aby testovací subjekt musel informaci chvíli hledat. Rozložení informací a jednotlivých sekcí je na obou stránkách totožné, až na pozici grafu. Na stránce s šablonou GrayScale je graf v levé části a na šabloně Freelancer je v části pravé (viz. obrázek 3.8).



Obrázek 3.8: Ukázka testovacích stránek (v prostřední časti je zobrazen graf a jeho popis, podle kterého testovaní uživatelé určovali, kterou verzi produktu stáhnout). Stránka zobrazená vlevo používá šablonu GrayScale a stránka vpravo šablonu Freelancer.

Protože je kladen důraz na sémantiku stránek, je k vytvoření grafu je použita technologie SVG , která naproti HTML 5 Canvas elementu poskytuje více detailní rozdělení na jednotlivé části grafu při sledování a to je vhodnější pro následné analyzování dat. Ze stejného důvodu jsou jednotlivé části rozděleny na adekvátně pojmenované sekce, navigaci a patičku. Jde tedy vhodný o příklad stránek pro následný rozbor.

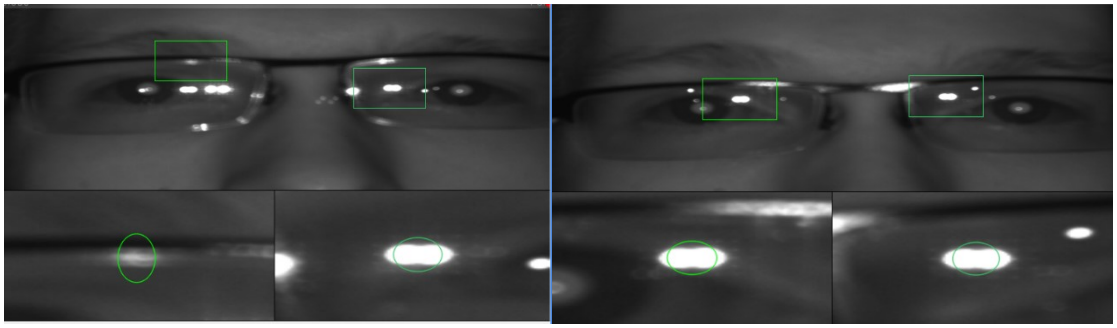
Po provedení všech požadovaných úkonů byl každý testovaný uživatel dotázán, kterou stránku preferoval z hlediska vzhledu a použitelnosti. Při měření bylo každému uživateli doporučeno co nejrychlejší splnění úkolu.

3.4.3 Souhrn informací získaných z testování řešení

Testování řešení bylo prováděno na sedmi uživatelích. Každý provedl zadané úkoly na obou připravených webových stránkách. Pořadí v jakém bylo testování prováděno, bylo u všech testovaných uživatelů stejné. Uživatelé začínali s webovou stránkou v tématu GreyScale (tmavý design) a pokračovali se stránkou v tématu Freelancer (světlý design). Je tak možné získat náhled na zlepšení konkrétních uživatelů mezi stránkami a zároveň porovnávat uživatele mezi sebou. Záporem tohoto přístupu je, že porovnání výsledků z první a druhé stránky je obtížnější a nemusí být tak vypovídající, protože vzhled stránek je podobný a uživatel má tak u druhé stránky nějaké zkušenosti s tím jak postupovat. Jiným potencionálním řešením, pro zachování prvního dojmu uživatele by bylo testování na stránkách, které by se podstatně lišily ve vzhledu i rozložení ovládacích prvků.

3.4.3.1 ***Problémy objevené při testování***

Během testování se projevilo několik problémů. Největším byl problém s funkčností zařízení pro měření pozice očí. Pokud měl testovaný uživatel nasazeny dioptrické brýle, zařízení nebylo schopno ve většině případech udržet pozici očí. S pomocí kalibrace a pohybu kamery se dařilo (někdy po několika minutách nastavování) přimět zařízení k nalezení pozice očí, ale při jakémkoli mírném pohybu bylo zaměření ztraceno. Dle mých testů se se zvyšujícími se dioptriemi brýlí, mírně zhoršuje funkčnost zařízení (usuzuji tak pouze z malého počtu vzorků, protože testy probíhaly na příliš malém počtu uživatelů a navíc neprobíhaly ve stejných světelných podmínkách). Tyto problémy byly mírně tlumeny v případech, kdy byly brýle opatřeny vrstvou minimalizující odrazy od skel (tj. antireflexní úprava skel). Podobné chování se objevovalo jak při přirozeném (slunečním) tak i umělém světle, při dobrých světelných podmínkách i při podmínkách zhoršených. Hlavním důvodem těchto problémů je častá neschopnost zařízení rozlišit mezi očima uživatele a odleskem na sklech, případně na obroučkách brýlí. Odlesky byly způsobeny buď zdrojem světla v místnosti, nebo přímo monitorem měřeného počítače. Zmenšením jasu monitoru počítače jsem byl schopen v některých případech zvýšit schopnost zařízení nalézt oči uživatele, ovšem tím byly zhoršeny rozpoznávací schopnosti uživatele - uživatelé často nebyli schopni dostatečně dobře číst displej. Ačkoli výrobce uvádí, že zařízení je schopno fungovat i pokud má měřený uživatel nasazeny dioptrické brýle, z prováděných měření se zdálo, že tato funkce pravděpodobně funguje jen při velice specifických podmínkách. S použitím větší obrazovky se přesnost měření nadále zvyšovala a zařízení fungovalo podstatně lépe, i když měl měřený uživatel nasazené brýle. Při měřeních s použitím 24" monitoru jsem byl schopen dosáhnout znatelně lepší přesnosti měření než při použití displeje notebooku (15,6"). Pokud zde zmiňuji přesnost, myslím tím převážně počet validních dat, které se mi podařilo získat za měřený časový úsek. Je potřeba brát v potaz, že testování přesnosti měření jako takové je velice složité, protože pokud jsou nepřesnosti malé, uživatel je nemusí sám rozpoznat, nelze tedy jednoduše rozlišit nepřesnost od skutečné validní hodnoty.



Obrázek 3.9: Ukázky problémů s odlesky na brýlích. Zvýrazněné oblasti naznačují místa, kde zařízení hledalo oči uživatele.

Některé z testů byly tímto problémem ovlivněny až v průběhu měření, to znamená, že zařízení ztratilo zaměření očí ve chvíli plnění úkolu (uživatel už tedy viděl testovanou stránku a byl seznámen s jeho úkoly). Takováto měření jsem byl nucen považovat za neplatné. Tyto chybné měření se projevovaly především při spuštění aplikace pro podporu analýzy, kde při zobrazení pohybu uživatele chybělo navštívení oblastí stránky, které byly nutné pro dokončení testu (každý testovaný uživatel požadovaný úkol dokončil). Už při porovnání doby měření a počtu získaných dat bylo zřejmé, že podstatná část záznamů chybí (v jednom případě až 50%). Pokud například z prvního a posledního záznamu jsem schopen zjistit, že měření trvalo 20 vteřin, ale součet trvání (duration) v naměřených hodnotách dává hodnotu 15 vteřin, můžu usoudit, že až 25% dat potencionálně chybí. Předpokládám totiž, že uživatel po dobu provádění testu sledoval monitor (z mého pozorování uživatele). Některé takto vadné měření byly provedeny vícekrát. Mnohonásobným měřením stejné stránky nebylo umožněno analyzovat uživatelův první dojem z navštívených stránek. Pomocí možnosti simulování pohybu očí kurzorem myši jsem se ujistil v tom, že nedostatek dat nebyl způsoben chybou aplikace, která při tomto nastavení fungovala bez problémů.

3.4.3.2 ***Reakce na vyskytnuté problémy***

Měření některých uživatelů se nepodařilo ani po několika pokusech a tak jsem byl nucen počet měřených uživatelů omezit (uživatele, kteří nebyli zařízením nalezeni, ani po několika pokusech byli z měření vyřazeni). Protože měření je prováděno převážně z důvodu ověření funkčností měření, neměl by menší počet měření znamenat podstatný problém. Po tomto omezení v testu zůstalo celkem pět uživatelů.

3.4.4 Výsledky měření

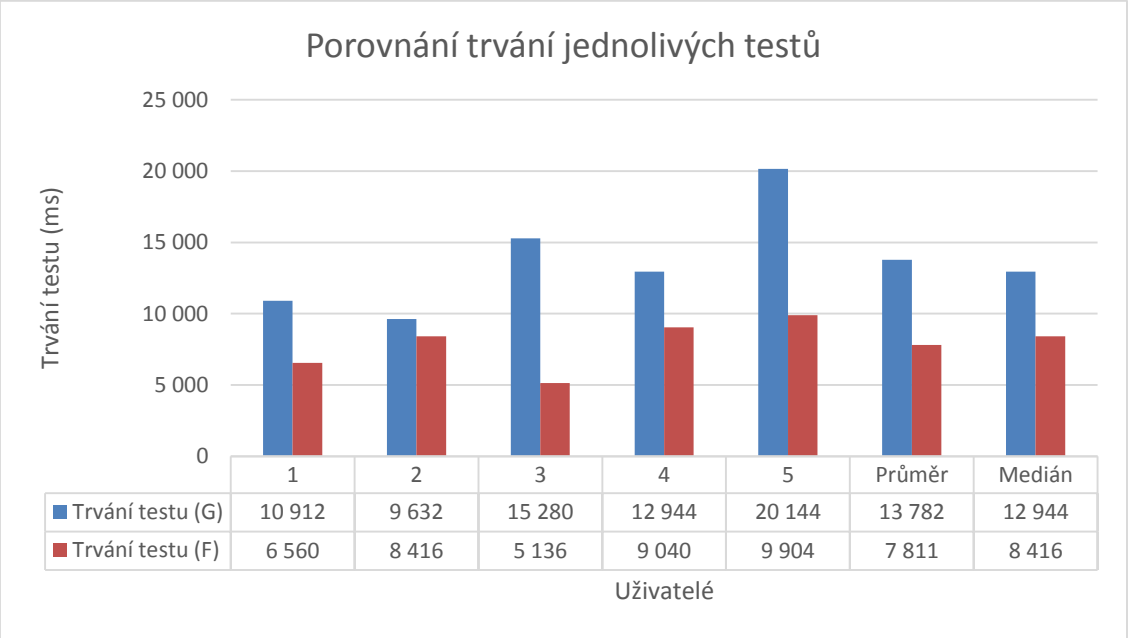
Tato kapitola poskytuje pohled na samotný rozbor a analýzu získaných informací. Některé z grafů v této části nejsou vygenerovány aplikací, ale vytvořeny jinými nástroji pomocí dat z aplikace. Výhodou tohoto postupuje větší flexibilita a snadnější vizuální porovnání výsledků.

3.4.4.1 ***Pohled na výsledky jako na celek***

Čtyři z pěti testovaných uživatelů preferovali šablonu s tmavšími barvami (šablona Greyscale) a umístěním grafu před jeho popisem (tzn. Graf vlevo a popis vpravo) a jenom jeden testovaný uživatel pak preferoval stránku s šablonou Freelancer (světlá šablona). Stránka s šablonou Greyscale byla u všech uživatelů měřena jako první.

Tabulka 3.2: Hodnoty z měření jednotlivých testů (G je zkratka pro šablonu Greyscale a F pro šablonu Freelancer)

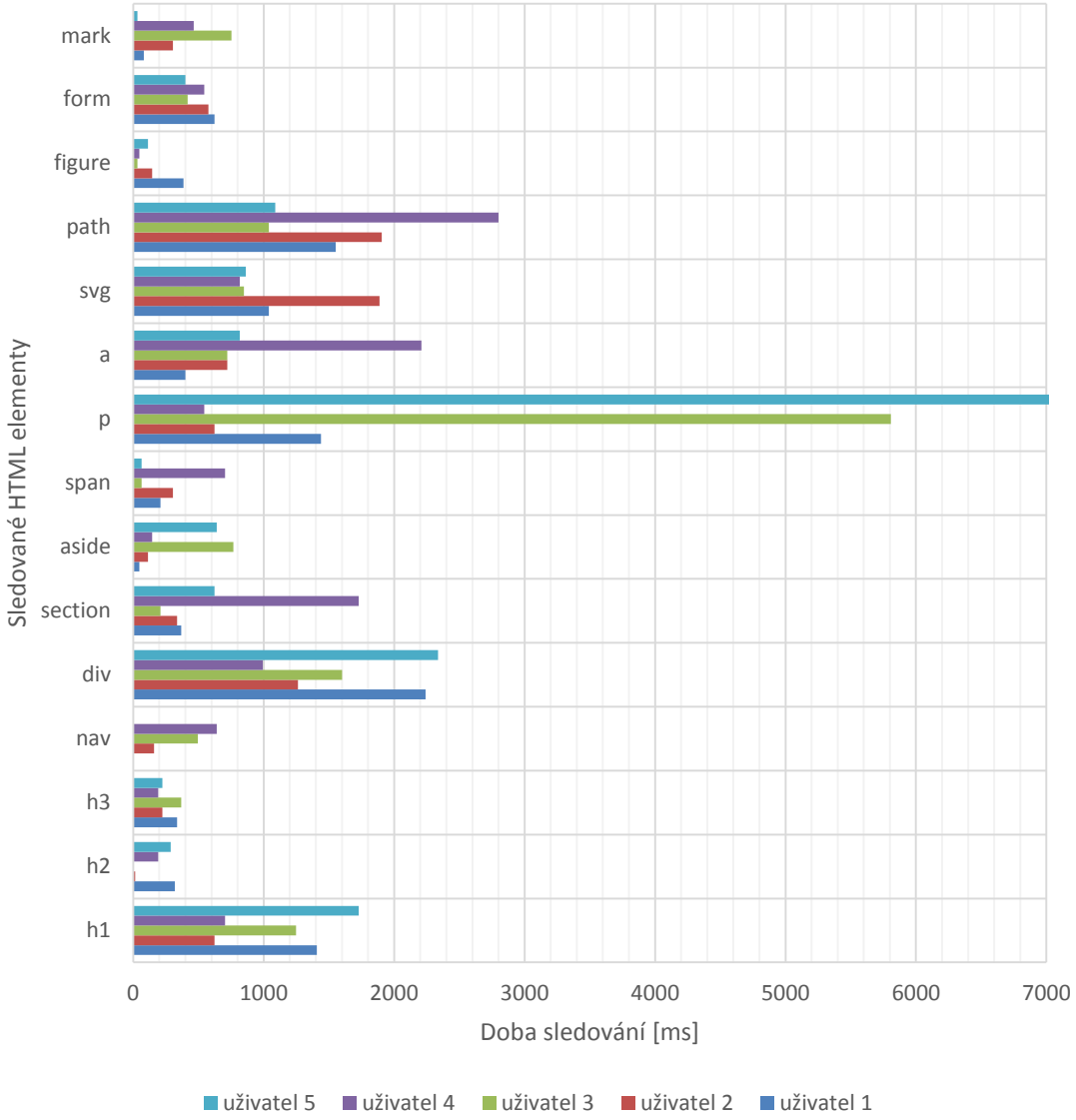
Uživatel	1	2	3	4	5	Průměr	Medián
Záznamů (G)	676	598	945	803	1 243	853	803
Trvání testu (G)[ms]	10 912	9 632	15 280	12 944	20 144	13 782	12 944
Záznamů (F)	409	520	321	562	612	485	520
Trvání testu (F)[ms]	6 560	8 416	5 136	9 040	9 904	7 811	8 416
Zlepšení (%)	40	13	66	30	51	40	40
Preference šablony	G	G	G	F	G		



Obrázek 3.10: Graf s porovnaným trváním jednotlivých testů jak v rámci uživatele, tak mezi jednotlivými uživateli. U všech měřených uživatelů došlo po navštívení první stránky ke zlepšení orientace na stránce, takové zlepšení bylo očekáváno vzhledem k podobnosti testovacích stránek.

Mezi prvním a druhým měřením lze pozorovat zrychlení orientace jednotlivých uživatelů. Průměrná hodnota i hodnota mediánu tohoto zlepšení činila 40%. Uživatelé, kteří u

kterých první test trval kratší dobu, nedosáhli tak výrazného zlepšení jako uživatelé, kterým první měření trvalo déle. To bylo zřejmě způsobeno tím, že se doba jejich testu více blížila nejkratší možné době nalezení požadovaných informací a splnění zadaného úkolu - nebylo tedy tolik prostoru ke zlepšení. Jednotlivá měření trvala průměrně 13 789ms u prvního měření, při druhém testu pak průměrně 7 811ms. Průměrné pozorované zlepšení 40% znamená zrychlení o 5 971ms. Nejdelší a nejkratší měření se od sebe liší o 9 632ms u prvního měření a o 5 136 u měření druhého. Tyto výsledky jsou v souladu s tím, že během prvního měření někteří uživatelé hledali ovládací prvky uživatelského rozhraní a orientovali se v celkovém rozložení stránek, kdežto při druhém testu už měli zkušenost z prvního měření a tak jejich orientace trvala kratší dobu. Hodnoty zlepšení, průměrů a mediánu jsou zaokrouhleny na celé čísla, protože v tomto případě nemá význam zavádět hodnoty větší přesnosti.



Obrázek 3.11: Graf poměru doby sledování jednotlivých elementů u měřených uživatelů.

Obrázek 3.11 zobrazuje rozdíly mezi dobami sledování jednotlivých HTML elementů u měřených uživatelů. Graf je zjednodušený o elementy, které měly příliš malou hodnotu. Je to z důvodu zvýšení čitelnosti tohoto grafu. Konkrétně jde o elementy h4, h5, i, line, ul, th, td, time, label, input a footer. Ze stejného důvodu je omezena i maximální hodnota u vodorovné osy na 7s. Celková hodnota doby sledování p elementu u pátého uživatele je 9 904ms. Takováto hodnota jasně značí, že si uživatel pročítal přiložený popis grafu. To je nejdelší text na testované stránce, popisující vývoj popularity jednotlivých verzí za poslední léta.

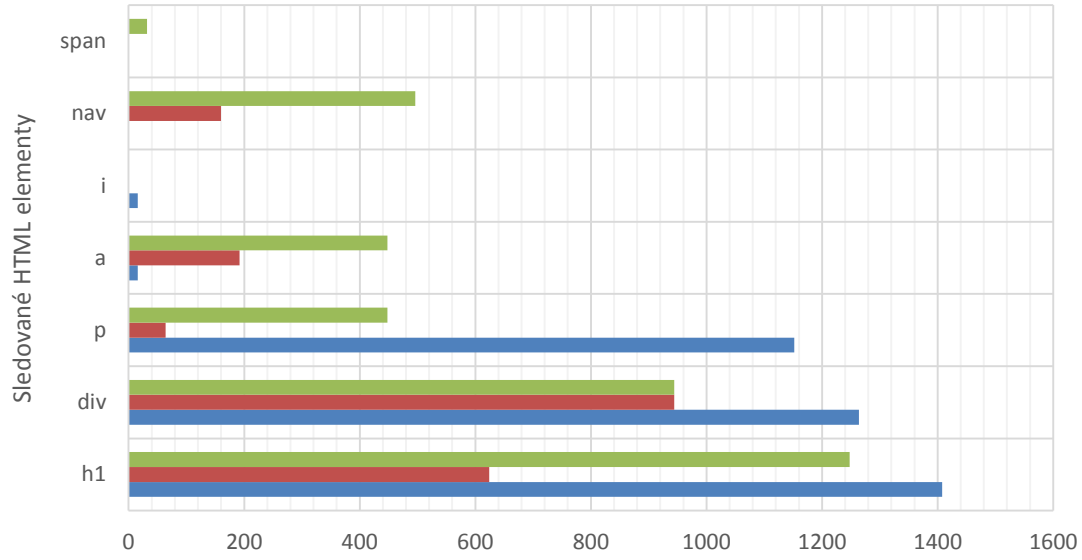
Kratší doby sledování elementů svg, path a line, značí, že uživatelé se rychleji orientují v grafu než v popisném textu. Text také poskytoval potřebné informace pro splnění úkolu. Pro ověření ale všichni testovaní uživatelé sledovali i graf a to včetně těch, kteří text sledovali nejdéle (uživatel 5 a uživatel 3). Lze proto předpokládat, že pro většinu uživatelů, je reprezentace dat je vhodnější pomocí grafu, než pomocí popisného textu. Někteří uživatelé využili text pouze jako legendu k zobrazenému grafu.

Z grafů jako je tento (Obrázek 4.5) lze přečíst hodně informací, ale už z prvního pohledu je jasné, že v mnoha případech bude lepší data rozdělit do částí. Měření je možné rozdělit podle času, podle zajímavých částí stránky nebo podle jednotlivých úkolů uživatele.

3.4.4.2 ***Rozdělení výsledků na části***

Pro lepší srozumitelnost výsledných dat jsem rozdělil výsledky jednotlivých měření do tří logických celků. Protože v dokumentu je obtížné dobře znázornit všechny výsledky, nadále budu pracovat se pouze se třemi z pěti uživatelů.

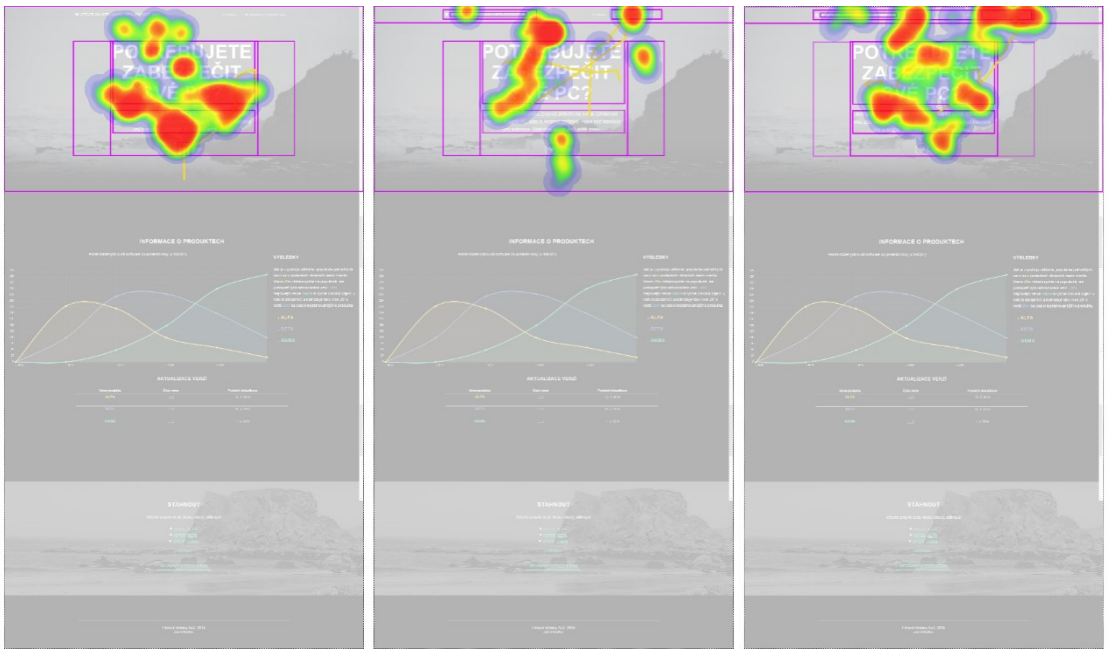
- První částí je orientace na stránce, hledání ovládacích prvků nebo menu. První sekce začíná u všech měření v době 0ms a končí, jakmile uživatel nalezne část s informacemi o produktech.
- Druhá část končí, jakmile uživatel provede samotné získávání informací pro vybrání správného produktu.
- Poslední část začíná koncem té předchozí a končí stažením správné verze (ukončení měření).



	h1	div	p	a	i	nav	span
uživatel 3	1248	944	448	448	0	496	32
uživatel 2	624	944	64	192	0	160	0
uživatel 1	1408	1264	1152	16	16	0	0

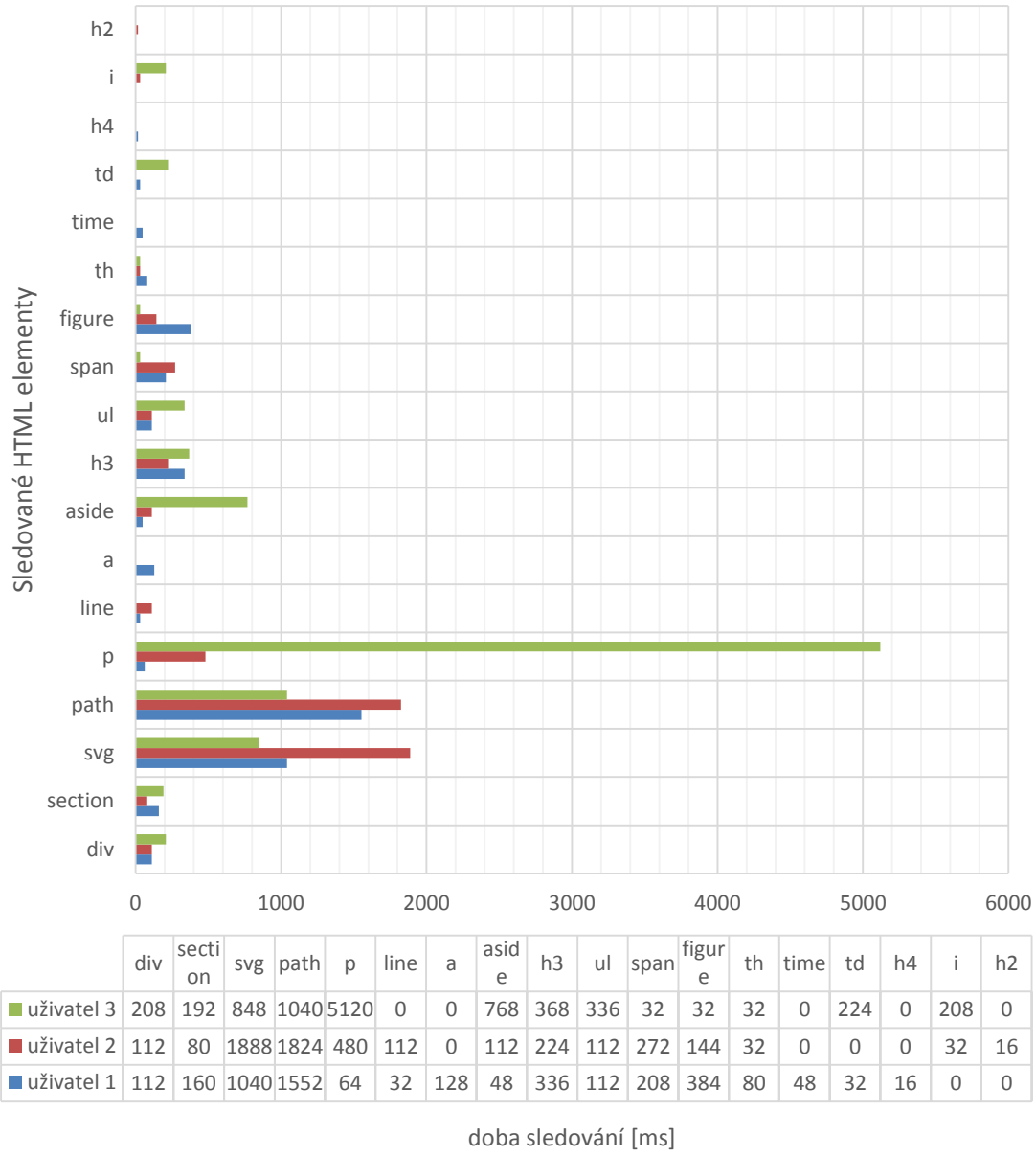
doba sledování [ms]

Obrázek 3.12: Na tomto grafu jsou zobrazeny výsledky měření uživatelů 1, 2 a 3 na šabloně Greyscale z první části stránky.



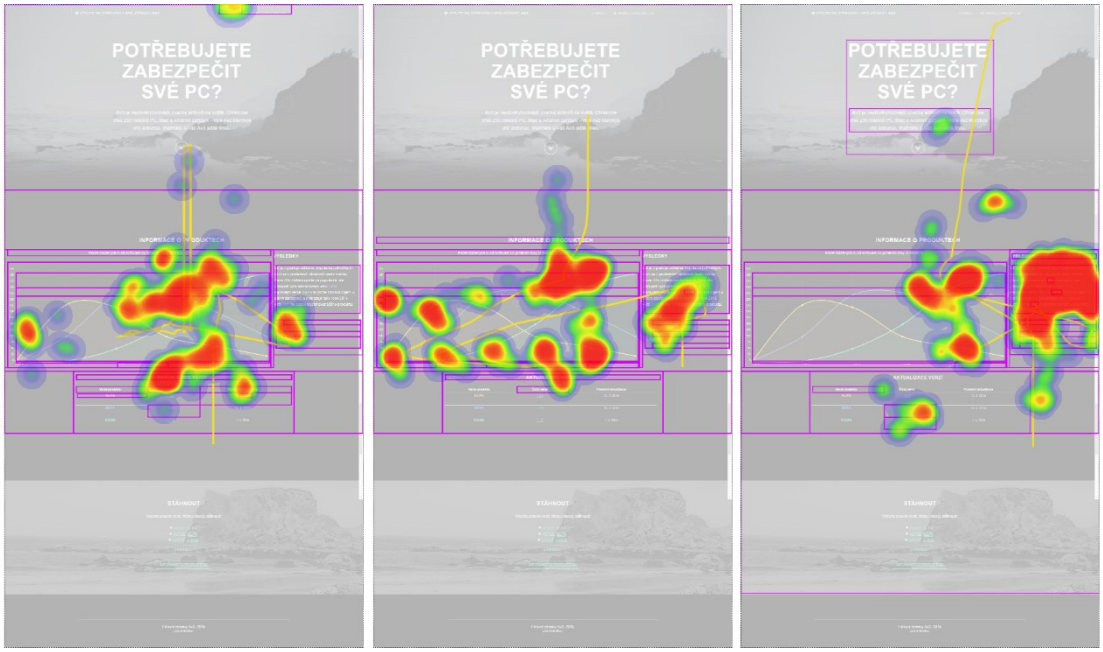
Obrázek 3.13: Heatmapy měření jednotlivých uživatelů z prvních částí.

První část webové stránky se skládá především z velkého nadpisu (h1) a textu pod ním. V zobrazené části stránky je také navigace, která je zobrazená i na ostatních částech měření. Z grafu sledovaných elementů (obrázek 3.12) i získaných heatmap (obrázek 3.13) lze vidět mírně odlišný přístup testovaných uživatelů. Uživatel 1 se zabýval většinu času čtením hlavního nadpisu a doprovodného textu, kdežto uživatel 2 se textu téměř nevěnoval (pouze 688ms) a pokračoval rovnou odkazem v menu. Třetí uživatel sice textu pozornost věnoval, ale následně také pokračoval tlačítkem v menu. Uživatel 1 zvolil tlačítko u textu místo menu, což je zřetelné jak z příslušné heatmapy tak z absence sledování prvku nav (navigace) a odkazů.



	div	section	svg	path	p	line	a	aside	h3	ul	span	figure	th	time	td	h4	i	h2
uživatel 3	208	192	848	1040	5120	0	0	768	368	336	32	32	32	0	224	0	208	0
uživatel 2	112	80	1888	1824	480	112	0	112	224	112	272	144	32	0	0	0	32	16
uživatel 1	112	160	1040	1552	64	32	128	48	336	112	208	384	80	48	32	16	0	0

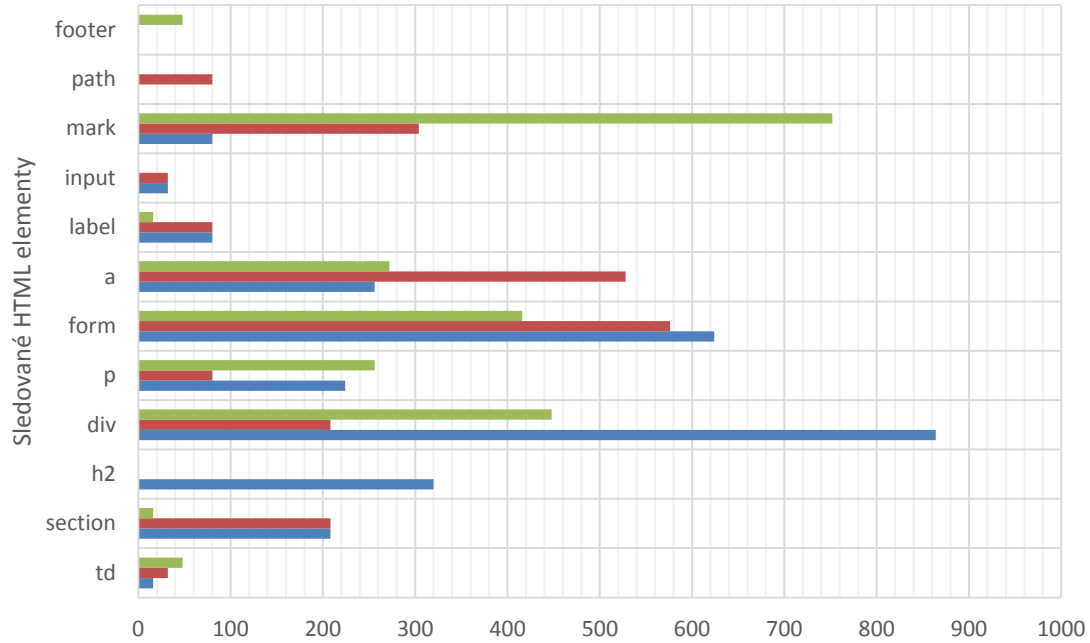
Obrázek 3.14: Výsledky měření uživatelů 1, 2 a 3 na šabloně Greyscale z druhé části stránky. Tento graf obsahuje nejvíce dat, jde o tu část stránky, kde uživatelé čtou potřebné informace.



Obrázek 3.15: Heatmapy uživatelů 1, 2 a 3 (zleva) pro druhou část měřené stránky.

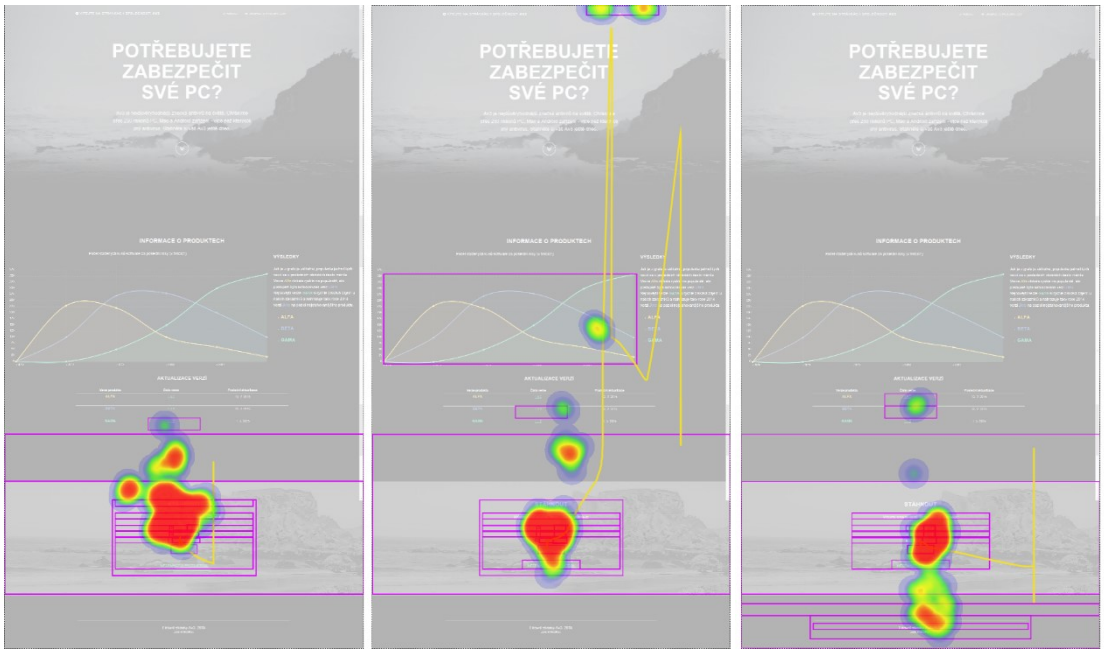
Ve druhé části stránky uživatelé strávili nejdelší část testu. Nejzajímavější je tu porovnání mezi hledáním v grafu a přiloženém textu. Jak je zřetelné z obrázků 3.14 a 3.15 uživatel 1 hledal všechny informace v grafu, což se odráží nejenom na snímku heatmapy ale i na poměrně dlouhém sledování elementu svg a path, které tvoří graf. Příslušnou legendu uživatel vyhledal až následně. Podobně postupoval i uživatel 2, i když jeho chování nebylo tak rychlé a zaměřil tak ještě více na graf. Zajímavější je chování třetího uživatele, který grafovou část webovou stránky sledoval relativně málo, v poměru s dobou sledování přidruženého textu. O četbě textu vypovídá převážně více než 5s sledování elementu p a aside (kontejner pro text).

Pod grafem byla vytvořena tabulka neobsahující žádné užitečné informace. Této tabulce se žádný z těchto tří uživatelů příliš nevěnoval, což je viditelné z krátkého sledování tabulkových elementů jako jsou th a td.



	td	section	h2	div	p	form	a	label	input	mark	path	footer
uživatel 3	48	16	0	448	256	416	272	16	0	752	0	48
uživatel 2	32	208	0	208	80	576	528	80	32	304	80	0
uživatel 1	16	208	320	864	224	624	256	80	32	80	0	0

Obrázek 3.16: Naměřená data pro třetí část měření šablony greyscale, pro uživatele 1,2 a 3.

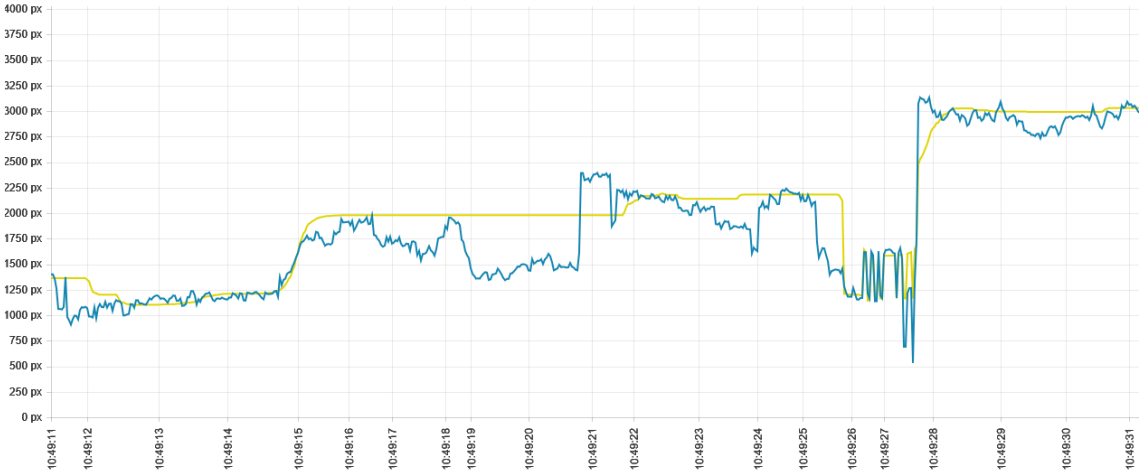


Obrázek 3.17: Porovnání heatmap z měření jednotlivých uživatelů (1,2,3), pro třetí část měření.

Na třetí části (obrázky 3.16 a 3.17) je viditelné například mírné sledování patičky stránky třetím uživatelem. Není překvapivé, že nejdelší čas uživatelé strávili sledováním elementů form, a a mark, protože tyto elementy tvořily jednotlivé možnosti stažení. Mark byl na testovacích stránkách použitý přímo pro vyznačení stahované verze. Jeho vysoký poměr v pozornosti uživatele značí, že tento prvek zaujal pozornost uživatelů (byl nastylován tučným podtrženým textem a navíc to byl kritický prvek pro ukončení testovaného scénáře). Uživatel 1 na tomto prvku moc času nestrávil, ale za to je vidět velký čas sledování div elementu. Tento element byl použit jako kontejner pro výběr verze, lze proto předpokládat, že to bylo způsobeno nepřesností měření.

3.4.4.3 Použití vztahu myši a pozice očí

Aplikace nabízí graf porovnávající pozici očí a kurzoru myši v čase. Z tohoto grafu lze přečíst různé informace o kmitání pohledu a rolování stránek. Lze získat náhled i na uživatelovu cestu stránkou, protože pokud se například často vrací na začátek stránky, bude to v grafu viditelné kmitáním křivky.

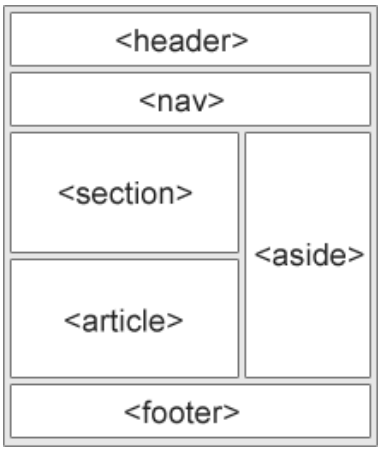


Obrázek 3.18: Ukázka z měření uživatel 4 na šabloně greyscale. Z grafu je viditelné. Postupně uživatel postupoval stránkou od shora dolů. Zhruba v 80% grafu je viditelný pokles obou křivek. Tento pokles byl způsoben zarolováním směrem nahoru. Mohlo tomu být například proto, že uživatel si potřeboval znova ověřit nějaké informace a vrátil se zpět. Druhou možností by bylo hledání potřebného ovládacího prvku či odkazu. Z grafu je také viditelné, že pochyby oka (modrá křivka) mají tendenci se velice rychle a často pohybovat, oproti tomu pohyb myši je ve většině času plynulý. I z tohoto důvodu je webová analýza s využitím eye trackingu přidanou hodnotou k analýze s využitím pouze myši (tzn., dodává více konkrétní informace).

3.4.4.4 Zhodnocení použitelnosti a výsledků

Ze sledovaných elementů je možné získat cenné informace pro další analýzu chování uživatele při průchodu webových stránek. Aby zobrazené grafy a heatmapy poskytovaly, co nejvíce informací je důležité znát i zdrojový kód a sémantiku měřené stránky. Různé kontejnerové elementy typu DIV, SPAN, SECTION atd. mohou vnést nepříznivé rušení do

vykreslených grafů, je proto vhodné počítat s tím že tyto elementy mají velice různorodou funkčnost a nenesou konkrétní informace (převážně element DIV). Oproti tomu některé z nových sémantických elementů specifikace HTML5 jako jsou například elementy ARTICLE, HEADER, FIGURE mohou nést více konkrétní informace o svém obsahu.



Obrázek 3.19: Možné uspořádání HTML 5 sémantických elementů k dosažení lepších informací z prováděných měření.

Některé nepřesnosti mohou být způsobeny i samotným měřením, obzvláště u malých elementů, jako jsou různé zvýrazňovací prvky, je toto chování běžné. Je proto nutné sledovat i element, který je kontejnerem pro daný prvek.

Grafy z celého měření mají tendenci být nepřehledné a je proto výhodné si je rozdělit do několika bloků, podle očekávaných zajímavých oblastí webové stránky. Pro testování jenom konkrétní části stránky, je vhodné vyfiltrovat si data podle požadované pozice v DOM (například podle sekce stránky). Ve výsledcích se pak lze zbavit znečištění grafu prvky, které pro testování nejsou zajímavé.

Při správném použití představuje kombinace eye trackingu se sémantikou stránky a klasickou webovou analýzou mocný nástroj pro analýzu chování uživatele na webových stránkách. Po dostatečně velkém množství testů lze informace získané z tohoto řešení využít i pro návrh vylepšení pro uživatelské rozhraní.

4 Budoucí práce a možné využití

4.1 Použití v reálných aplikacích

Využití sledování pozice očí společně se související sémantikou zdrojového kódu pro analýzu webových stránek a uživatelského rozhraní má význam převážně v situacích, kde jsou správně využity sémantické HTML elementy. Pokud jsou použity běžné a obecné elementy div (případně span) na veškerou strukturu webové stránky, hrozí riziko, že vygenerované grafy nebudou nést dostatečnou informaci. Elementy obsažené ve svých kontejnerech totiž často nezaujímají celou velikost svého rodičovského elementu nebo ji přesahují (většinou u absolutně pozicovaných elementů). Použití správného obsahového elementu podle jeho obsahu tedy vyplní chybějící informace, kdežto obecné obsahové elementy rozeznatelné nebudou.

Využití tohoto řešení je tedy výhodnější v nových aplikacích, převážně jednostránkového typu. Správná funkčnost i výpovědnost aplikace by měla být poskytnuta i u takzvaných SPA a RIA aplikací implementovaných například s pomocí frameworků angular.js, ember.js nebo jiných. Při použití těchto aplikací je nutné vždy v aplikaci pro analýzu zvolit správný časový či sémantický úsek. Zobrazení všech dat najednou bude vést k nepřiměřené komplikovanosti vygenerovaných grafů. Konkrétní použití je ale individuální podle měřené aplikace, možností je spousta.

Použití je možné i pro zkoumání důležitých nosných částí stránek. Pokud jsou data správně vyfiltrována podle struktury konkrétního HTML, je možné na výsledky pohlížet stejně, jako by se jednalo o celou webovou aplikaci.

4.2 Použití pro návrh uživatelského rozraní

Potencionál využití tohoto řešení pro návrh či analýzu uživatelského rozhraní je zřejmý. V kombinaci s klasickými metodami analýzy webových stránek lze toto řešení využít při měření chování testovacích uživatelů. Nejlépe takových, kteří jsou budoucími (nebo současnými) zákazníky a uživateli implementované aplikace.

S využitím vygenerované struktury sledovaných elementů lze identifikovat takové části zdrojového kódu, které nebyly žádným uživatelem sledovány a jsou tady pravděpodobně zbytečné nebo umístěné nesprávně. Naopak elementy upoutávající uživatelovu pozornost příliš oproti jejich záměru, je taktéž možno identifikovat v adekvátních grafech. Takové elementy nebo části stránek lze pak umístit jinam nebo změnit jejich vzhled natolik aby k podobným nedorozuměním u uživatelů nedocházelo.

Vizuální reprezentaci zobrazovaných elementů lze porovnat s pozadím webové stránky a pozicemi pohledů očí. V takovém případě je umožněna identifikace částí stránek, které nedostávají adekvátní poměr uživatelovy pozornosti nebo nejsou sledovány vůbec.

S využitím grafu s pozicemi pohledu očí a kurzoru myši je možné identifikovat uživatelovo zmatení ovládacími prvky nebo jeho časté hledání na webové stránce. V takových

případech je možné například doporučit změnu rozvržení UI nebo přidání dalších detailnějších odkazů.

Implementované řešení není jediným nástrojem pro návrh a mělo by sloužit převážně jako nástroj k identifikaci možných problémů a nesrovnalostí v použití konkrétních webových stránek. Využití sledování pozice očí společně se související sémantikou pro analýzu webových stránek a uživatelského rozhraní má tedy význam převážně v situacích, kde jsou správně využity sémantické elementy. Pokud jsou použity běžné a obecné elementy div na veškerou strukturu webové stránky, hrozí riziko, že vygenerované grafy nebudou nést dostatečné informace (jako je popsáno v předešlých kapitolách). To je ale individuální chování, které je potřeba zvážit z hlediska očekávaného výsledku. Závěry vyvozené z provedené analýzy by měly být vždy otestovány na dostatečně velkém a vypovídajícím počtu testovaných uživatelů.

4.3 Prostor pro zlepšení

V každé aplikaci existuje prostor pro zlepšení, či alternativní přístup k problému. Ani toto řešení není výjimkou. Jedním z takových vylepšení by mohlo být použití jiného zařízení pro sledování pozice pohledu, aby bylo možno dosáhnout větší přesnosti měření a zaměření pohledu (i při stavu kdy měřený uživatel nosí silnější dioptrické brýle). Provedení podobných měření s použitím jiného zařízení by také zvýšilo věrohodnost získaných dat a důvěru v přesnost a kvalitu naměřených výsledků. Vždy totiž existuje možnost, že používané zařízení obsahuje nějakou chybu způsobující nepřesnost, ať už je tato chyba softwarového nebo hardwarového typu. Tato eventuální změna zařízení je možná i beze změny použitého řešení za předpokladu, že zařízení bude posílat data ve stejném formátu s využitím stejné síťové technologie (v opačném případě lze použít jednoduchou formátovací aplikaci a stále pokračovat se zbytkem současného řešení).

Pokud pominu zařízení, potencionální vylepšení by mohlo nabídnout i použití frameworku pro komunikaci pomocí protokolu websocket (od doby implementace mého řešení se objevily některá řešení obsahující pokročilejší funkce, které má implementace neobsahuje). S větší složitostí nejspíše dojde k pomalejšímu běhu, takže i jednoduší naimplementované řešení má své výhody. Další možnou nevýhodou je využití řešení s více aplikacemi. Implementace, která by vše sjednocovala do jedné aplikace (nejspíše by šlo o dedikovaný prohlížeč) by nabídnula jednoduší používání a funkce, které řešením rozděleném na čtyři aplikace dosáhnout nelze.

V případě potřeby se nabízí i doplnění aplikace pro analýzu o podpůrné funkce pro export dat do formátu CSV nebo přímo do obrázků. Dalším vylepšením pro měření rozsáhlých aplikací s velkým množstvím různých URL adres by bylo filtrování zobrazovaných dat pomocí jejich přiřazené adresy, podobně jako je tomu u filtrace podle jejich umístění ve struktuře DOM. Pro některé uživatele by také mohla být zajímavá možnost využití konfiguračních souborů a podobných vylepšení uživatelské zkušenosti s programem.

Závěr

V práci se mi povedlo naimplementovat funkční řešení, které slouží jako nástroj pro analýzu webových stránek s použitím možností eye trackingu, umožněného s pomocí zařízení Gazepoint GP3 Eye Tracker. Výsledek klade důraz na sémantickou část měřených webových stránek a tím je schopen poskytnout detailnější informace. Kompletní řešení je rozděleno do tří aplikací, rozdělených dle jejich oblasti funkcionality. Data změřené zařízením pro eye tracking jsou zpracovávána v reálném čase aplikací zastávající úlohu přeposílacího serveru. Tato aplikace data transformuje, zpracovává a přeposílá skriptu, který běží na stránce a propojuje získaná data s aktuálním děním na analyzované webové stránce. Pro snadnou analýzu změřených dat slouží dedikovaná aplikace nabízející různé vizualizace a filtrace dat. Funkčnost a validitu řešení jsem testoval s pomocí několika vytvořených testovacích stránek. Měření bylo prováděno na pěti testovacích uživatelích, s tím že každý provedl dva různé testovací případy. Výsledky testů ukázaly potencionál tohoto přístupu k webové analýze, pokud má osoba provádějící analýzu dostatečnou znalost o zdrojovém kódu respektive sémantice testované webové stránky. Oproti klasickým metodám tato kombinace nabízí více podrobné data a s využitím možností filtrování podle času a sémantiky, je možné zaměřit se i na malé části webových stránek.

Provedené testy a měření naznačují mimo jiné použitelnost pro identifikaci jak strukturálních oblasti stránky, kterým se nedostává očekávané pozornosti uživatele, tak takových oblastí které naopak příliš odtahují pozornost uživatele od toho co je na stránce důležité. Takovéto informace jsou často zajímavé pro návrh vylepšení uživatelského rozhraní webové stránky.

Práce byla použita pro zpracování článku s tématem analýzy syntaktických elementů a struktury webových stránek pomocí technologie eye tracking [5]. Tento článek sloužil jako příspěvek ke konferenci ICC (tj. International Carpathian Control Conference), na kterou byl následně přijat.

Použitá literatura

[1] H. Kenneth a M. Nyström, Eye tracking: a comprehensive guide to methods and measures, New York: Oxford University Press, 2011.

[2] A. Bulling, J. A. Ward, H. Gellersen a G. Tröster, „Eye Movement Analysis for Activity Recognition Using Electrooculography," Cambridge, Lancaster, Zurich, 2011.

[3] A. A. Faisal a W. W. Abbott, „Ultra-low-cost 3D gaze estimation: an intuitive high information throughput compliment to direct brain–machine interfaces," Londýn, 2012.

[4] M. Radecký a P. Smutný, „Evaluating User Reaction to User Interface Element Using Eye-Tracking Technology," VŠB-Technical University of Ostrava, Ostrava.

[5] M. Radecký, J. Vykopal a P. Smutný, „Analysis of Syntactic Elements and Structure of Web Pages Using Eye-tracking Technology," VŠB-Technical University of Ostrava, Ostrava, 2015.

[6] A. Duchowski, „Eye Tracking Methodology: Theory and Practice," 2007.

[7] J. R. Bergstrom, „Eye Tracking the UX of Mobile: What You Need to Know," [Online]. Available: http://www.slideshare.net/JenniferRomanoBergstrom/eye-tracking-the-ux-of-mobile-what-you-need-to-know. [Přístup získán 2015].

[8] P. Philip S. Holzman, M. Leonard R. Proctor, D. L. Levy, N. J. Yasillo, M. Herbert Y. Meltzer a S. W. Hurt, „Eye-Tracking Dysfunctions in Schizophrenic Patients and Their Relatives," Arch Gen Psychiatry, 1974.

[9] F. Cullinan, „How to look for content clues in your analytics," 2011. [Online]. Available: http://firehead.net/2011/04/how-to-look-for-content-clues-in-your-analytics/. [Přístup získán 2015].

[10] R. Allen, „Web Analytics: What Is It Good For?," 2012. [Online]. Available: http://meetcontent.com/blog/web-analytics-what-is-it-good-for/. [Přístup získán 2015].

[11] A. Melnikov, „The WebSocket Protocol," 2011.

[12] J. Woods, „PPP Deflate Protocol," 1996.

[13] Ó. Marbán, G. Mariscal a J. Segovia, „A Data Mining & Knowledge Discovery Process Model," Madrid, 2009.

Součástí DP je CD.

Adresářová struktura přiloženého CD:

- Zdroje
  - Obrazky (obrázky a snímky obrazovky, pro řešení a dokument)
  - Eye_soubory (naměřené hodnoty)
  - Excel (excel soubory s použitými daty a grafy)
- Zdrojove_kody
  - Preposilaci_aplikace
    - Src (zdrojové soubory řešení)
    - Release (spustitelné řešení přeposílací aplikace)
  - Skript_a_testovaci_reseni
    - Src (řešení s testovacími stránkami a skriptem)
    - Skript (výsledký skript pro měření)
  - Aplikace_pro_analyzu (zdroje pro windows 8 aplikaci pro analýzu)
    - Src (zdrojové soubory řešení)
    - Package (instalovatelný balíček windows 8 aplikace)